Doxygen Book

Generated by Doxygen 1.8.17

Chapter 1

nnstreamer

1.1 Introduction

- Introduction : Neural Network Streamer for AI Projects

1.2 Program Name

- Program Name : nnstreamer
- Program Details : It provides a neural network framework connectivities (e.g., tensorflow, caffe) for gstreamer streams. Efficient Streaming for AI Projects: Neural network models wanted to use efficient and flexible streaming management as well. Intelligent Media Filters!: Use a neural network model as a media filter / converter. Composite Models!: Allow to use multiple neural network models in a single stream instance. Multi Model Intelligence!: Allow to use multiple sources for neural network models.

1.3 Input/output data

- INPUT : None
- OUTPUT : None

1.4 Code information

- Initial date : 2018/06/14
- Version : 0.1

Chapter 2

Todo List

Member _backtrace_to_string (void)

The .c file location of this function might be not appropriate.

Member _get_filenames (nnsconf_type_path type, const gchar *dir, GSList **listF, GSList **listN, guint *counter)

This assumes .so/.dylib for all sub plugins. Support Windows!

Member _gst_tensor_debug_output (GstTensorDebug *self, GstBuffer *buffer)

NYI: do the debug task

Member _GstTensorDecoder::tensor_config

support tensors in the future

Member _strdup_getenv (const gchar *name)

Evaluate if we need to use secure_getenv() here (and compatible with other OS

Member _validate_file (nnsconf_type_path type, const gchar *fullpath)

how to validate with nnsconf type.

Member ACCL_NPU

Define ACCL_DSP

Member ACCL_NPU_STR

Define ACCL_DSP_STR

Member custom_loadlib (const GstTensorFilterProperties *prop, void **private_data)

: Check the integrity of filter->data and filter->model_file, nnfw

Double check if this check is really required and safe

Member EVENT_NAME_UPDATE_MODEL

rename & move this to better location

Member g_tensor_filter_single_invoke (GTensorFilterSingle *self, const GstTensorMemory *input, GstTensorMemory *output, gboolean allocate)

refactor this local variable

how can we remove memcpy if output data is already allocated single-shot should fill the output data, but sub-plugin allocates new memory.

Member gst_mqtt_sink_set_host_address (GstMqttSink *self, const gchar *addr)

Handle the case where the addr is changed at runtime

Member gst_mqtt_sink_start (GstBaseSink *basesink)

Support other persistence mechanisms MQTTCLIENT_PERSISTENCE_NONE: A memory-based persistence mechanism MQTTCLIENT_PERSISTENCE_DEFAULT: The default file system-based persistence mechanism MQTTCLIENT_PERSISTENCE_USER: An application-specific persistence mechanism

Member gst_mqtt_src_create (GstBaseSrc *basesrc, guint64 offset, guint size, GstBuffer **buf)

DEFAULT_MQTT_SUB_TIMEOUT_MIN is too long

If the difference between new latency and old latency, gst_element_post_message (GST_ELEMENT_CAST (self), gst_message_new_latency (GST_OBJECT_CAST (self))); is needed.

: Send EoS here

Member gst_mqtt_src_set_host_address (GstMqttSrc *self, const gchar *addr)

Handle the case where the addr is changed at runtime

Member gst_mqtt_src_start (GstBaseSrc *basesrc)

Support other persistence mechanisms MQTTCLIENT_PERSISTENCE_NONE: A memory-based persistence mechanism MQTTCLIENT_PERSISTENCE_DEFAULT: The default file system-based persistence mechanism MQTTCLIENT_PERSISTENCE_USER: An application-specific persistence mechanism

Member gst_tensor_aggregator_chain (GstPad *pad, GstObject *parent, GstBuffer *buf)

flush data Invalid state, tried to flush large size. We have to determine how to handle this case. (flush the out-size or all available bytes) Now all available bytes in adapter will be flushed.

Member gst_tensor_aggregator_parse_caps (GstTensorAggregator *self, const GstCaps *caps)

flush data Check properties to detect invalid case. Assertion when in=5 out=10 flush=20 or in=10 out=5 flush=20

Member gst_tensor_buffer_append_memory (GstBuffer *buffer, GstMemory *memory, const GstTensorInfo *info)

Make custom gst_allocator later?

Member gst_tensor_converter_chain (GstPad *pad, GstObject *parent, GstBuffer *buf)

need rewrite. do not use assert

expand mem if given property is larger than mem size. Now compare same size, later we should modify mem block if developer sets different dimension.

identify and printout the given input stream caps.

Member gst_tensor_converter_parse_video (GstTensorConverter *self, const GstCaps *caps, GstTensorsConfig *config)

Add more conditions!

need rewrite.

Member gst_tensor_converter_set_property (GObject *object, guint prop_id, const GValue *value, GParamSpec *pspec)

detects framework based on the script extension

Member gst_tensor_converter_video_stride (GstVideoFormat format, gint width)

The actual list is much longer, fill them. (read https://gstreamer.freedesktop.org/documentation/design/ html)

Member gst_tensor_crop_do_cropping (GstTensorCrop *self, GstBuffer *raw, tensor_crop_info_s *cinfo)

Add various mode to crop tensor.

Member gst_tensor_crop_get_crop_info (GstTensorCrop *self, GstBuffer *info, tensor_crop_info_s *cinfo)

Add various mode to crop tensor. Now tensor-crop handles NHWC data format only.

Member gst_tensor_debug_class_init (GstTensorDebugClass *klass)

check the behavior of name and nick (output methods vs output)

Member gst_tensor_filter_configure_tensor (GstTensorFilter *self, const GstCaps *incaps)

We do not support this (flexible tensor for flexible input model). Cap-negotiation of the current tensor-filter requires either side of "model / set-property" or "incoming gstcaps" to be static/explicit. Ideally, this should support flexible tensor for flexible input model, leaving the negotiation to other elements, but we didn't implement it yet.

framerate of output tensors How can we update the framerate? GstTensorFilter cannot assure the framerate. Simply set the framerate of out-tensor from incaps.

Member gst_tensor_filter_transform_caps (GstBaseTransform *trans, GstPadDirection direction, GstCaps *caps, GstCaps *filter)

how to set the framerate of output tensors

We do not have a testcase hitting here. Thus, we do not ensure the validity here. However, according to gstreamer doxygen entry, if filter is given, that's not to be ignored. For now, we assume that if caps-size is 0, filter is "ANY".

Member gst_tensor_info_convert_to_meta (GstTensorInfo *info, GstTensorMetaInfo *meta)

handle rank from info.dimension

Member gst_tensor_meta_info_parse_header (GstTensorMetaInfo *meta, gpointer header)

update meta info for each version

Member gst_tensor_query_client_set_property (GObject *object, guint prop_id, const GValue *value, GParamSpec *pspec)

DO NOT update properties (host, port, ..) while pipeline is running.

Member gst_tensor_sparse_dec_chain (GstPad *pad, GstObject *parent, GstBuffer *buf)

consider more error handling

Member gst_tensor_trainer_set_prop_framework (GstTensorTrainer *trainer, const GValue *value)

Check valid framework

Member gst_tensor_transform_arithmetic (GstTensorTransform *filter, GstTensorInfo *in_info, GstTensorInfo *out_info, const uint8_t *inptr, uint8_t *outptr)

add more options

Member gst_tensor_transform_dimchg (GstTensorTransform *filter, GstTensorInfo *in_info, GstTensorInfo *out_info, const uint8_t *inptr, uint8_t *outptr)

do "IP" operation

CRITICAL-TODO: Optimize the performance!

NYI

Member gst_tensor_transform_padding (GstTensorTransform *filter, GstTensorInfo *in_info, GstTensorInfo *out_info, const uint8_t *inptr, uint8_t *outptr)

Add constant option instead of using zero padding value

Member gst_tensor_transform_transform (GstBaseTransform *trans, GstBuffer *inbuf, GstBuffer *outbuf)

max rank supported in tensor-transform is 4

Member gst_tensor_transform_transform_caps (GstBaseTransform *trans, GstPadDirection direction, GstCaps *caps, GstCaps *filtercap)

Get to know what the heck is @filtercap and use it!

Member gst_tensordec_transform_caps (GstBaseTransform *trans, GstPadDirection direction, GstCaps *caps, GstCaps *filter)

We may do more specific actions here

Member gst_tensordec_transform_size (GstBaseTransform *trans, GstPadDirection direction, GstCaps *caps, gsize size, GstCaps *othercaps, gsize *othersize)

If direction = SRC, you may need different interpretation!

Member gst_tensorsinfo_compare_print (const GstTensorsInfo *info1, const GstTensorsInfo *info2)

If this is going to be used by other elements, move this to nnstreamer/tensor_common.

File gsttensor_converter.c

For flatbuffers, support other/tensors with properties

Subplugins are not tested, yet.

File gsttensor_if.h

Add "event/signal" to reload FILL_WITH_FILE* file??? (TBD)

File gsttensor_srciio.c

support specific channels as input

handle timestamp received from device

Member mlagent_get_model_path_from (const GValue *val)

The specification of the data layout filled in the third argument (i.e., stringified_json) by the callee is not fully decided.

Parse stringified_json to get the model's path

Member nnsconf_dump (gchar *str, gulong size)

Add more configuration values to dump.

File ntputil.c

Need to support caching and polling timer mechanism

Member register_subplugin (subpluginType type, const char *name, const void *data)

data out of scope at add

Member TDBG_OUTPUT_CIRCULARBUF

NYI NOT_SUPPORTED)

Member TDBG_OUTPUT_FILEWRITE

NYI NOT_SUPPORTED)

Class tensor_crop_info_s

Add various mode to crop tensor. Now tensor-crop handles NHWC data format only.

File tensor_filter.c

set priority among properties

logic for dynamic properties(like model change)

File tensor_filter.h

TBD: Should we disable "in-place" mode? (what if output size > input size?)

Chapter 3

Bug List

File edge_common.c

No known bugs

File edge_common.h

No known bugs

File edge_elements.c

No known bugs

File edge_sink.c

No known bugs

File edge_sink.h

No known bugs

File edge_src.c

No known bugs

File edge_src.h

No known bugs

File gstdatarepo.c

No known bugs except for NYI items

File gstdatarepo.h

No known bugs except for NYI items

File gstdatareposink.c

No known bugs except for NYI items

No known bugs except for NYI items

File gstdatareposrc.c

No known bugs except for NYI items

File gstdatareposrc.h

No known bugs except for NYI items

File gstjoin.c

No known bugs except for NYI items

File gstjoin.h

No known bugs except for NYI items

File gsttensor_aggregator.c

No known bugs except for NYI items

File gsttensor_aggregator.h

No known bugs except for NYI items

File gsttensor_converter.c

No known bugs except for NYI items

File gsttensor_converter.h

No known bugs except for NYI items

File gsttensor_converter_media_info_audio.h

No known bugs except for NYI items

File gsttensor_converter_media_info_video.h

No known bugs except for NYI items

File gsttensor_converter_media_no_audio.h

No known bugs except for NYI items

File gsttensor_converter_media_no_video.h

No known bugs except for NYI items

File gsttensor_crop.c

No known bugs except for NYI items

File gsttensor_crop.h

No known bugs except for NYI items

File gsttensor_debug.c

No known bugs except for NYI items

File gsttensor_debug.h

No known bugs except for NYI items

File gsttensor_decoder.c

gst_tensordec_transform_size () may be incorrect if direction is SINK.

If configured = TRUE, it holds TRUE until exit. What if configuration changes in run-time?

File gsttensor_decoder.h

No known bugs except for NYI items

File gsttensor_demux.c

No known bugs except for NYI items

File gsttensor_demux.h

No known bugs except for NYI items

File gsttensor_if.c

No known bugs except for NYI items

File gsttensor_if.h

No known bugs except for NYI items

File gsttensor_merge.c

No known bugs except for NYI items

File gsttensor_merge.h

No known bugs except for NYI items

File gsttensor_mux.c

No known bugs except for NYI items

File gsttensor_mux.h

No known bugs except for NYI items

File gsttensor_rate.c

No known bugs except for NYI items

File gsttensor_rate.h

No known bugs except for NYI items

File gsttensor_repo.c

No known bugs except for NYI items

File gsttensor_repo.h

No known bugs except for NYI items

File gsttensor_reposink.c

No known bugs except for NYI items

File gsttensor_reposink.h

No known bugs except for NYI items

File gsttensor_reposrc.c

No known bugs except for NYI items

File gsttensor_reposrc.h

No known bugs except for NYI items

File gsttensor_sink.c

No known bugs except for NYI items

File gsttensor_sink.h

No known bugs except for NYI items

File gsttensor_sparsedec.c

No known bugs except for NYI items

File gsttensor_sparsedec.h

No known bugs except for NYI items

File gsttensor_sparseenc.c

No known bugs except for NYI items

File gsttensor_sparseenc.h

No known bugs except for NYI items

File gsttensor_sparseutil.c

No known bugs except for NYI items

File gsttensor_sparseutil.h

No known bugs except for NYI items

File gsttensor_split.c

No known bugs except for NYI items

File gsttensor_split.h

No known bugs except for NYI items

File gsttensor_srciio.c

No known bugs except for NYI items

File gsttensor_srciio.h

No known bugs except for NYI items

File gsttensor_trainer.c

No known bugs except for NYI items

File gsttensor_trainer.h

No known bugs except for NYI items

File gsttensor_transform.c

This is NYI.

File gsttensor_transform.h

No known bugs.

File hw_accel.c

No known bugs except for NYI items

File hw_accel.h

No known bugs except for NYI items

File ml_agent.c

No known bugs except for NYI items

File ml_agent.h

No known bugs except for NYI items

File mqttcommon.h

No known bugs except for NYI items

File mqttsink.c

No known bugs except for NYI items

No known bugs except for NYI items

File mqttsink.h

No known bugs except for NYI items

File mqttsrc.c

No known bugs except for NYI items

File mqttsrc.h

No known bugs except for NYI items

File nnstreamer.c

No known bugs except for NYI items

File nnstreamer_conf.c

No known bugs except for NYI items

File nnstreamer_conf.h

No known bugs except for NYI items

File nnstreamer_internal.h

No known bugs except for NYI items

File nnstreamer_log.c

No known bugs except for NYI items

File nnstreamer_log.h

No known bugs except for NYI items

File nnstreamer_plugin_api.h

No known bugs except for NYI items

File nnstreamer_plugin_api_converter.h

No known bugs except for NYI items

File nnstreamer_plugin_api_decoder.h

No known bugs except for NYI items

File nnstreamer_plugin_api_filter.h

No known bugs except for NYI items

File nnstreamer_plugin_api_impl.c

No known bugs except for NYI items

File nnstreamer_plugin_api_trainer.h

No known bugs except for NYI items

File nnstreamer_plugin_api_util.h

No known bugs except for NYI items

File nnstreamer_plugin_api_util_impl.c

No known bugs except for NYI items

File nnstreamer_subplugin.c

No known bugs except for NYI items

File nnstreamer_subplugin.h

No known bugs except for NYI items

File nnstreamer_util.h

No known bugs except for NYI items

File nnstreamer_watchdog.c

No known bugs except for NYI items

File nnstreamer_watchdog.h

No known bugs except for NYI items

File ntputil.c

No known bugs except for NYI items

File ntputil.h

No known bugs except for NYI items

File tensor_allocator.c

No known bugs

File tensor_common.h

No known bugs except for NYI items

File tensor_converter_custom.h

No known bugs except for NYI items

File tensor_data.c

No known bugs except for NYI items

File tensor_data.h

No known bugs except for NYI items

File tensor_decoder_custom.h

No known bugs except for NYI items

File tensor_filter.c

No known bugs except for NYI items

File tensor_filter.h

No known bugs except for NYI items

File tensor_filter_common.c

No known bugs except for NYI items

File tensor_filter_common.h

No known bugs except for NYI items

File tensor_filter_custom.c

No known bugs except for NYI items

File tensor_filter_custom.h

No known bugs except for NYI items

File tensor_filter_custom_easy.c

No known bugs except for NYI items

File tensor_filter_custom_easy.h

No known bugs except for NYI items

File tensor_filter_single.c

No known bugs except for NYI items

File tensor_filter_single.h

No known bugs except for NYI items

File tensor_filter_support_cc.cc

No known bugs except for NYI items

File tensor_if.h

No known bugs except for NYI items

File tensor_meta.c

No known bugs

File tensor_meta.h

No known bugs

File tensor_query_client.c

No known bugs

File tensor_query_client.h

No known bugs

File tensor_query_common.c

No known bugs except for NYI items

File tensor_query_common.h

No known bugs except for NYI items

File tensor_query_server.c

No known bugs

File tensor_query_server.h

No known bugs

File tensor_query_serversink.c

No known bugs

File tensor_query_serversink.h

No known bugs

File tensor_query_serversrc.c

No known bugs

File tensor_query_serversrc.h

No known bugs

File tensor_typedef.h

No known bugs except for NYI items

Chapter 4

Namespace Index

4.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 5

Class Index

5.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 6

File Index

6.1 File List

Here is a list of all files with brief descriptions:

Chapter 7

Namespace Documentation

7.1 nnstreamer Namespace Reference

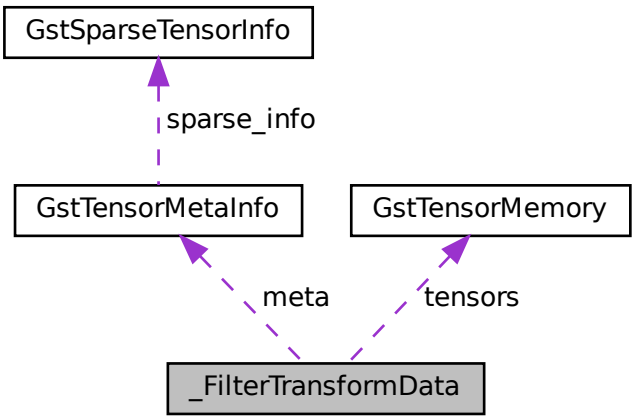
Chapter 8

Class Documentation

8.1 _FilterTransformData Struct Reference

Internal data structure for tensor_filter transform data.

Collaboration diagram for _FilterTransformData:



Public Attributes

- GstMemory * mem [NNS_TENSOR_SIZE_LIMIT]
- GstMapInfo info [NNS_TENSOR_SIZE_LIMIT]
- GstTensorMetaInfo meta [NNS_TENSOR_SIZE_LIMIT]
- guint num_tensors
- GstTensorMemory tensors [NNS_TENSOR_SIZE_LIMIT]
- gboolean is_flexible
- gboolean allocate_in_invoke

8.1.1 Detailed Description

Internal data structure for tensor_filter transform data.

Definition at line 153 of file tensor_filter.c.

8.1.2 Member Data Documentation

8.1.2.1 allocate_in_invoke

```
gboolean _FilterTransformData::allocate_in_invoke
```

Definition at line 162 of file tensor_filter.c.

8.1.2.2 info

```
GstMapInfo _FilterTransformData::info[NNS_TENSOR_SIZE_LIMIT]
```

Definition at line 156 of file tensor_filter.c.

8.1.2.3 is_flexible

```
gboolean _FilterTransformData::is_flexible
```

Definition at line 161 of file tensor_filter.c.

8.1.2.4 mem

```
GstMemory* _FilterTransformData::mem[NNS_TENSOR_SIZE_LIMIT]
```

Definition at line 155 of file tensor_filter.c.

8.1.2.5 meta

```
GstTensorMetaInfo _FilterTransformData::meta[NNS_TENSOR_SIZE_LIMIT]
```

Definition at line 157 of file tensor_filter.c.

8.1.2.6 num_tensors

```
guint _FilterTransformData::num_tensors
```

Definition at line 158 of file tensor_filter.c.

8.1.2.7 tensors

```
GstTensorMemory _FilterTransformData::tensors[NNS_TENSOR_SIZE_LIMIT]
```

Definition at line 159 of file tensor_filter.c.

The documentation for this struct was generated from the following file:

- nnstreamer/tensor_filter/tensor_filter.c

8.2 _GstDataRepoSink Struct Reference

GstDataRepoSink data structure.

```
#include <gstdatareposink.h>
```

Public Attributes

- GstBaseSink element
- GstCaps * fixed_caps
- JsonObject * json_object
- JsonArray * sample_offset_array
- JsonArray * tensor_size_array
- JsonArray * tensor_count_array
- guint cumulative_tensors
- gboolean is_static_tensors
- gint fd
- GstDataRepoDataType data_type
- gint total_samples
- guint64 fd_offset
- gsize sample_size
- gchar * filename
- gchar * json_filename

8.2.1 Detailed Description

GstDataRepoSink data structure.

Definition at line 40 of file gstdatareposink.h.

8.2.2 Member Data Documentation

8.2.2.1 cumulative_tensors

```
guint _GstDataRepoSink::cumulative_tensors
```

the number of cumulated tensors

Definition at line 49 of file gstdatareposink.h.

8.2.2.2 data_type

```
GstDataRepoDataType _GstDataRepoSink::data_type
```

data type

Definition at line 53 of file gstdatareposink.h.

8.2.2.3 element

```
GstBaseSink _GstDataRepoSink::element
```

Definition at line 42 of file gstdatareposink.h.

8.2.2.4 fd

```
gint _GstDataRepoSink::fd
```

open file descriptor

Definition at line 52 of file gstdatareposink.h.

8.2.2.5 fd_offset

```
guint64 _GstDataRepoSink::fd_offset
```

offset of fd

Definition at line 55 of file gstdatareposink.h.

8.2.2.6 filename

```
gchar* _GstDataRepoSink::filename
```

filename

Definition at line 59 of file gstdatareposink.h.

8.2.2.7 fixed_caps

```
GstCaps* _GstDataRepoSink::fixed_caps
```

to get meta info

Definition at line 44 of file gstdatareposink.h.

8.2.2.8 is_static_tensors

```
gboolean _GstDataRepoSink::is_static_tensors
```

Definition at line 51 of file gstdatareposink.h.

8.2.2.9 json_filename

```
gchar* _GstDataRepoSink::json_filename
```

"JSON file path to store the meta information

Definition at line 60 of file gstdatareposink.h.

8.2.2.10 json_object

```
JsonObject* _GstDataRepoSink::json_object
```

JSON object

Definition at line 45 of file gstdatareposink.h.

8.2.2.11 sample_offset_array

```
JsonArray* _GstDataRepoSink::sample_offset_array
```

offset array of sample

Definition at line 46 of file gstdatareposink.h.

8.2.2.12 sample_size

```
gsize _GstDataRepoSink::sample_size
```

size of one sample

Definition at line 56 of file gstdatareposink.h.

8.2.2.13 tensor_count_array

```
JsonArray* _GstDataRepoSink::tensor_count_array
```

array for the number of cumulative tensors

Definition at line 48 of file gstdatareposink.h.

8.2.2.14 tensor_size_array

```
JsonArray* _GstDataRepoSink::tensor_size_array
```

size array of flexible tensor

Definition at line 47 of file gstdatareposink.h.

8.2.2.15 total_samples

```
gint _GstDataRepoSink::total_samples
```

The number of total samples, in the case of multi-files, it is used as an index.

Definition at line 54 of file gstdatareposink.h.

The documentation for this struct was generated from the following file:

- datarepo/gstdatareposink.h

8.3 _GstDataRepoSinkClass Struct Reference

GstDataRepoSinkClass data structure.

`#include <gstdatareposink.h>`

Public Attributes

- GstBaseSinkClass parent_class

8.3.1 Detailed Description

GstDataRepoSinkClass data structure.

Definition at line 66 of file gstdatareposink.h.

8.3.2 Member Data Documentation

8.3.2.1 parent_class

`GstBaseSinkClass _GstDataRepoSinkClass::parent_class`

Definition at line 68 of file gstdatareposink.h.

The documentation for this struct was generated from the following file:

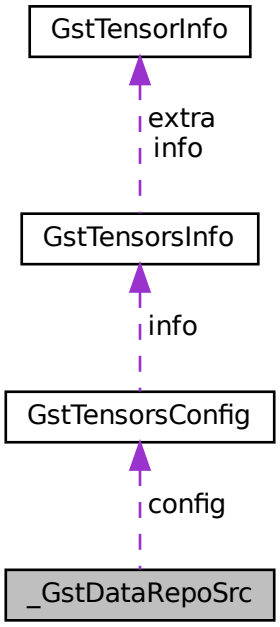
- datarepo/gstdatareposink.h

8.4 _GstDataRepoSrc Struct Reference

GstDataRepoSrc data structure.

`#include <gstdatareposrc.h>`

Collaboration diagram for _GstDataRepoSrc:



Public Attributes

- GstPushSrc parent
- GstPad * src_pad
- gboolean is_start
- gboolean successful_read
- gint fd
- gint file_size
- guint64 read_position
- guint64 fd_offset
- guint64 start_offset
- guint64 last_offset
- gsize tensors_size [NNS_TENSOR_SIZE_LIMIT]
- gsize tensors_offset [NNS_TENSOR_SIZE_LIMIT]
- gint current_sample_index
- gboolean first_epoch_is_done
- guint total_samples
- guint num_samples
- gsize sample_size
- GstDataRepoDataType data_type
- gchar * filename
- gchar * json_filename
- gchar * tensors_seq_str
- guint start_sample_index
- guint stop_sample_index

- guint epochs
- gboolean is_shuffle
- GArray * shuffled_index_array
- guint array_index
- guint tensors_seq [NNS_TENSOR_SIZE_LIMIT]
- guint tensors_seq_cnt
- gboolean need_changed_caps
- GstCaps * caps
- GstTensorsConfig config
- JsonArray * sample_offset_array
- JsonArray * tensor_size_array
- JsonArray * tensor_count_array
- JsonParser * parser
- guint sample_offset_array_len
- guint tensor_size_array_len
- guint tensor_count_array_len
- GstClockTime running_time
- gint rate_n
- gint rate_d
- guint64 n_frame

8.4.1 Detailed Description

GstDataRepoSrc data structure.

Definition at line 41 of file gstdatareposrc.h.

8.4.2 Member Data Documentation

8.4.2.1 array_index

```
guint _GstDataRepoSrc::array_index
```

element index of shuffled_index_array

Definition at line 73 of file gstdatareposrc.h.

8.4.2.2 caps

```
GstCaps* _GstDataRepoSrc::caps
```

optional property, datareposrc should get data format from JSON file caps field

Definition at line 78 of file gstdatareposrc.h.

8.4.2.3 config

```
GstTensorsConfig _GstDataRepoSrc::config
```

tensors information from current caps

Definition at line 81 of file gstdatareposrc.h.

8.4.2.4 current_sample_index

```
gint _GstDataRepoSrc::current_sample_index
```

current index of sample or file to read

Definition at line 56 of file gstdatareposrc.h.

8.4.2.5 data_type

```
GstDataRepoDataType _GstDataRepoSrc::data_type
```

media type

Definition at line 61 of file gstdatareposrc.h.

8.4.2.6 epochs

```
guint _GstDataRepoSrc::epochs
```

repetition of range of files or samples to read

Definition at line 69 of file gstdatareposrc.h.

8.4.2.7 fd

```
gint _GstDataRepoSrc::fd
```

open file descriptor

Definition at line 48 of file gstdatareposrc.h.

8.4.2.8 fd_offset

```
guint64 _GstDataRepoSrc::fd_offset
```

offset of fd

Definition at line 51 of file gstdatareposrc.h.

8.4.2.9 file_size

```
gint _GstDataRepoSrc::file_size
```

file size, in bytes

Definition at line 49 of file gstdatareposrc.h.

8.4.2.10 filename

```
gchar* _GstDataRepoSrc::filename
```

filename

Definition at line 64 of file gstdatareposrc.h.

8.4.2.11 first_epoch_is_done

```
gboolean _GstDataRepoSrc::first_epoch_is_done
```

Definition at line 57 of file gstdatareposrc.h.

8.4.2.12 is_shuffle

```
gboolean _GstDataRepoSrc::is_shuffle
```

shuffle the sample index

Definition at line 70 of file gstdatareposrc.h.

8.4.2.13 is_start

```
gboolean _GstDataRepoSrc::is_start
```

check if datareposrc is started

Definition at line 46 of file gstdatareposrc.h.

8.4.2.14 json_filename

```
gchar* _GstDataRepoSrc::json_filename
```

json filename containing meta information of the filename

Definition at line 65 of file gstdatareposrc.h.

8.4.2.15 last_offset

```
guint64 _GstDataRepoSrc::last_offset
```

last offset to read

Definition at line 53 of file gstdatareposrc.h.

8.4.2.16 n_frame

```
guint64 _GstDataRepoSrc::n_frame
```

Definition at line 92 of file gstdatareposrc.h.

8.4.2.17 need_changed_caps

```
gboolean _GstDataRepoSrc::need_changed_caps
```

When tensors-sequence changes, caps need to be changed

Definition at line 77 of file gstdatareposrc.h.

8.4.2.18 num_samples

```
guint _GstDataRepoSrc::num_samples
```

The number of samples to be used out of the total samples in the file

Definition at line 59 of file gstdatareposrc.h.

8.4.2.19 parent

```
GstPushSrc _GstDataRepoSrc::parent
```

parent object

Definition at line 43 of file gstdatareposrc.h.

8.4.2.20 parser

```
JsonParser* _GstDataRepoSrc::parser
```

Keep JSON data after parsing JSON file

Definition at line 85 of file gstdatareposrc.h.

8.4.2.21 rate_d

```
gint _GstDataRepoSrc::rate_d
```

Definition at line 91 of file gstdatareposrc.h.

8.4.2.22 rate_n

```
gint _GstDataRepoSrc::rate_n
```

Definition at line 91 of file gstdatareposrc.h.

8.4.2.23 read_position

```
guint64 _GstDataRepoSrc::read_position
```

position of fd

Definition at line 50 of file gstdatareposrc.h.

8.4.2.24 running_time

```
GstClockTime _GstDataRepoSrc::running_time
```

one frame running time

Definition at line 90 of file gstdatareposrc.h.

8.4.2.25 sample_offset_array

```
JsonArray* _GstDataRepoSrc::sample_offset_array
```

offset array of sample

Definition at line 82 of file gstdatareposrc.h.

8.4.2.26 sample_offset_array_len

```
guint _GstDataRepoSrc::sample_offset_array_len
```

Definition at line 86 of file gstdatareposrc.h.

8.4.2.27 sample_size

```
gsize _GstDataRepoSrc::sample_size
```

size of one sample

Definition at line 60 of file gstdatareposrc.h.

8.4.2.28 shuffled_index_array

```
GArray* _GstDataRepoSrc::shuffled_index_array
```

shuffled sample index array

Definition at line 72 of file gstdatareposrc.h.

8.4.2.29 src_pad

```
GstPad* _GstDataRepoSrc::src_pad
```

Definition at line 44 of file gstdatareposrc.h.

8.4.2.30 start_offset

```
guint64 _GstDataRepoSrc::start_offset
```

start offset to read

Definition at line 52 of file gstdatareposrc.h.

8.4.2.31 start_sample_index

```
guint _GstDataRepoSrc::start_sample_index
```

start index of sample to read, in case of image, the starting index of the numbered files

Definition at line 67 of file gstdatareposrc.h.

8.4.2.32 stop_sample_index

```
guint _GstDataRepoSrc::stop_sample_index
```

stop index of sample to read, in case of image, the stoppting index of the numbered files

Definition at line 68 of file gstdatareposrc.h.

8.4.2.33 successful_read

```
gboolean _GstDataRepoSrc::successful_read
```

used for checking EOS when reading more than one images(multi-files) from a path

Definition at line 47 of file gstdatareposrc.h.

8.4.2.34 tensor_count_array

```
JsonArray* _GstDataRepoSrc::tensor_count_array
```

array for the number of cumulative tensors

Definition at line 84 of file gstdatareposrc.h.

8.4.2.35 tensor_count_array_len

```
guint _GstDataRepoSrc::tensor_count_array_len
```

Definition at line 88 of file gstdatareposrc.h.

8.4.2.36 tensor_size_array

```
JsonArray* _GstDataRepoSrc::tensor_size_array
```

size array of flexible tensor to be stored in a Gstbuffer

Definition at line 83 of file gstdatareposrc.h.

8.4.2.37 tensor_size_array_len

```
guint _GstDataRepoSrc::tensor_size_array_len
```

Definition at line 87 of file gstdatareposrc.h.

8.4.2.38 tensors_offset

```
gsize _GstDataRepoSrc::tensors_offset[NNS_TENSOR_SIZE_LIMIT]
```

each tensors offset in a sample

Definition at line 55 of file gstdatareposrc.h.

8.4.2.39 tensors_seq

```
guint _GstDataRepoSrc::tensors_seq[NNS_TENSOR_SIZE_LIMIT]
```

tensors sequence in a sample that will be read into gstbuffer

Definition at line 75 of file gstdatareposrc.h.

8.4.2.40 tensors_seq_cnt

```
guint _GstDataRepoSrc::tensors_seq_cnt
```

Definition at line 76 of file gstdatareposrc.h.

8.4.2.41 tensors_seq_str

```
gchar* _GstDataRepoSrc::tensors_seq_str
```

tensors in a sample are read into gstBuffer according to tensors_sequence

Definition at line 66 of file gstdatareposrc.h.

8.4.2.42 tensors_size

```
gsize _GstDataRepoSrc::tensors_size[NNS_TENSOR_SIZE_LIMIT]
```

each tensors size in a sample

Definition at line 54 of file gstdatareposrc.h.

8.4.2.43 total_samples

```
guint _GstDataRepoSrc::total_samples
```

The number of total samples

Definition at line 58 of file gstdatareposrc.h.

The documentation for this struct was generated from the following file:

- datarepo/gstdatareposrc.h

8.5 _GstDataRepoSrcClass Struct Reference

GstDataRepoSrcClass data structure.

```
#include <gstdatareposrc.h>
```

Public Attributes

- GstPushSrcClass parent_class

8.5.1 Detailed Description

GstDataRepoSrcClass data structure.

Definition at line 98 of file gstdatareposrc.h.

8.5.2 Member Data Documentation

8.5.2.1 parent_class

```
GstPushSrcClass _GstDataRepoSrcClass::parent_class
```

Definition at line 99 of file gstdatareposrc.h.

The documentation for this struct was generated from the following file:

- datarepo/gstdatareposrc.h

8.6 _GstEdgeSink Struct Reference

GstEdgeSink data structure.

`#include <edge_sink.h>`

Public Attributes

- GstBaseSink element
- gchar * host
- guint16 port
- gchar * dest_host
- guint16 dest_port
- gchar * topic
- nns_edge_connect_type_e connect_type
- nns_edge_h edge_h
- gboolean wait_connection
- guint64 connection_timeout
- gchar * custom_lib
- gboolean is_connected
- GMutex lock
- GCond cond

8.6.1 Detailed Description

GstEdgeSink data structure.

Definition at line 42 of file edge_sink.h.

8.6.2 Member Data Documentation

8.6.2.1 cond

`GCond _GstEdgeSink::cond`

Definition at line 60 of file edge_sink.h.

8.6.2.2 connect_type

`nns_edge_connect_type_e _GstEdgeSink::connect_type`

Definition at line 52 of file edge_sink.h.

8.6.2.3 connection_timeout

```
guint64 _GstEdgeSink::connection_timeout
```

Definition at line 55 of file edge_sink.h.

8.6.2.4 custom_lib

```
gchar* _GstEdgeSink::custom_lib
```

Definition at line 57 of file edge_sink.h.

8.6.2.5 dest_host

```
gchar* _GstEdgeSink::dest_host
```

Definition at line 48 of file edge_sink.h.

8.6.2.6 dest_port

```
guint16 _GstEdgeSink::dest_port
```

Definition at line 49 of file edge_sink.h.

8.6.2.7 edge_h

```
nns_edge_h _GstEdgeSink::edge_h
```

Definition at line 53 of file edge_sink.h.

8.6.2.8 element

```
GstBaseSink _GstEdgeSink::element
```

Definition at line 44 of file edge_sink.h.

8.6.2.9 host

```
gchar* _GstEdgeSink::host
```

Definition at line 46 of file edge_sink.h.

8.6.2.10 is_connected

```
gboolean _GstEdgeSink::is_connected
```

Definition at line 58 of file edge_sink.h.

8.6.2.11 lock

```
GMutex _GstEdgeSink::lock
```

Definition at line 59 of file edge_sink.h.

8.6.2.12 port

```
guint16 _GstEdgeSink::port
```

Definition at line 47 of file edge_sink.h.

8.6.2.13 topic

```
gchar* _GstEdgeSink::topic
```

Definition at line 50 of file edge_sink.h.

8.6.2.14 wait_connection

```
gboolean _GstEdgeSink::wait_connection
```

Definition at line 54 of file edge_sink.h.

The documentation for this struct was generated from the following file:

- edge/edge_sink.h

8.7 _GstEdgeSinkClass Struct Reference

GstEdgeSinkClass data structure.

`#include <edge_sink.h>`

Public Attributes

- GstBaseSinkClass parent_class

8.7.1 Detailed Description

GstEdgeSinkClass data structure.

Definition at line 66 of file edge_sink.h.

8.7.2 Member Data Documentation

8.7.2.1 parent_class

`GstBaseSinkClass _GstEdgeSinkClass::parent_class`

parent class

Definition at line 68 of file edge_sink.h.

The documentation for this struct was generated from the following file:

- edge/edge_sink.h

8.8 _GstEdgeSrc Struct Reference

GstEdgeSrc data structure.

`#include <edge_src.h>`

Public Attributes

- GstBaseSrc element
- gchar * dest_host
- guint16 dest_port
- gchar * topic
- nns_edge_connect_type_e connect_type
- nns_edge_h edge_h
- GAsyncQueue * msg_queue
- gboolean playing
- gchar * custom_lib

8.8.1 Detailed Description

GstEdgeSrc data structure.

Definition at line 42 of file edge_src.h.

8.8.2 Member Data Documentation

8.8.2.1 connect_type

```
nns_edge_connect_type_e _GstEdgeSrc::connect_type
```

Definition at line 50 of file edge_src.h.

8.8.2.2 custom_lib

```
gchar* _GstEdgeSrc::custom_lib
```

Definition at line 56 of file edge_src.h.

8.8.2.3 dest_host

```
gchar* _GstEdgeSrc::dest_host
```

Definition at line 46 of file edge_src.h.

8.8.2.4 dest_port

```
guint16 _GstEdgeSrc::dest_port
```

Definition at line 47 of file edge_src.h.

8.8.2.5 edge_h

```
nns_edge_h _GstEdgeSrc::edge_h
```

Definition at line 51 of file edge_src.h.

8.8.2.6 element

```
GstBaseSrc _GstEdgeSrc::element
```

Definition at line 44 of file edge_src.h.

8.8.2.7 msg_queue

```
GAsyncQueue* _GstEdgeSrc::msg_queue
```

Definition at line 52 of file edge_src.h.

8.8.2.8 playing

```
gboolean _GstEdgeSrc::playing
```

Definition at line 54 of file edge_src.h.

8.8.2.9 topic

```
gchar* _GstEdgeSrc::topic
```

Definition at line 48 of file edge_src.h.

The documentation for this struct was generated from the following file:

- edge/edge_src.h

8.9 _GstEdgeSrcClass Struct Reference

GstEdgeSrcClass data structure.

```
#include <edge_src.h>
```

Public Attributes

- GstBaseSrcClass parent_class

8.9.1 Detailed Description

GstEdgeSrcClass data structure.

Definition at line 62 of file edge_src.h.

8.9.2 Member Data Documentation

8.9.2.1 parent_class

```
GstBaseSrcClass _GstEdgeSrcClass::parent_class
```

Definition at line 64 of file edge_src.h.

The documentation for this struct was generated from the following file:

- edge/edge_src.h

8.10 _GstJoin Struct Reference

Internal data structure for join instances.

```
#include <gstjoin.h>
```

Public Attributes

- GstElement element
- GstPad * srcpad
- GstPad * active_sinkpad
- guint n_pads
- guint padcount
- gboolean have_group_id
- GMutex lock
- GCond cond

8.10.1 Detailed Description

Internal data structure for join instances.

Definition at line 36 of file gstjoin.h.

8.10.2 Member Data Documentation

8.10.2.1 active_sinkpad

```
GstPad* _GstJoin::active_sinkpad
```

Definition at line 42 of file gstjoin.h.

8.10.2.2 cond

```
GCond _GstJoin::cond
```

Definition at line 49 of file gstjoin.h.

8.10.2.3 element

```
GstElement _GstJoin::element
```

Definition at line 38 of file gstjoin.h.

8.10.2.4 have_group_id

```
gboolean _GstJoin::have_group_id
```

Definition at line 46 of file gstjoin.h.

8.10.2.5 lock

```
GMutex _GstJoin::lock
```

Definition at line 48 of file gstjoin.h.

8.10.2.6 n_pads

```
guint _GstJoin::n_pads
```

Definition at line 43 of file gstjoin.h.

8.10.2.7 padcount

```
guint _GstJoin::padcount
```

Definition at line 44 of file gstjoin.h.

8.10.2.8 srcpad

```
GstPad* _GstJoin::srcpad
```

Definition at line 40 of file gstjoin.h.

The documentation for this struct was generated from the following file:

- join/gstjoin.h

8.11 _GstJoinClass Struct Reference

GstJoinClass inherits GstElementClass.

```
#include <gstjoin.h>
```

Public Attributes

- GstElementClass parent_class

8.11.1 Detailed Description

GstJoinClass inherits GstElementClass.

Definition at line 55 of file gstjoin.h.

8.11.2 Member Data Documentation

8.11.2.1 parent_class

```
GstElementClass _GstJoinClass::parent_class
```

Definition at line 57 of file gstjoin.h.

The documentation for this struct was generated from the following file:

- join/gstjoin.h

8.12 _GstJoinPad Struct Reference

GstJoinPad data structure.

Public Attributes

- GstPad parent
- guint group_id
- GstSegment segment
- guint32 segment_seqnum

8.12.1 Detailed Description

GstJoinPad data structure.

Definition at line 93 of file gstjoin.c.

8.12.2 Member Data Documentation

8.12.2.1 group_id

```
guint _GstJoinPad::group_id
```

Definition at line 97 of file gstjoin.c.

8.12.2.2 parent

```
GstPad _GstJoinPad::parent
```

Definition at line 95 of file gstjoin.c.

8.12.2.3 segment

```
GstSegment _GstJoinPad::segment
```

Definition at line 99 of file gstjoin.c.

8.12.2.4 segment_seqnum

```
guint32 _GstJoinPad::segment_seqnum
```

Definition at line 100 of file gstjoin.c.

The documentation for this struct was generated from the following file:

- join/gstjoin.c

8.13 _GstJoinPadClass Struct Reference

_GstJoinPadClass data structure

Public Attributes

- GstPadClass parent

8.13.1 Detailed Description

_GstJoinPadClass data structure

Definition at line 106 of file gstjoin.c.

8.13.2 Member Data Documentation

8.13.2.1 parent

```
GstPadClass _GstJoinPadClass::parent
```

Definition at line 108 of file gstjoin.c.

The documentation for this struct was generated from the following file:

- join/gstjoin.c

8.14 _GstMQTTMessageHdr Struct Reference

Defined a custom data type, GstMQTTMessageHdr.

```
#include <mqttcommon.h>
```

Public Attributes

- union {
  struct {
  guint num_mems
  gsize size_mems [GST_MQTT_MAX_NUM_MEMS]
  gint64 base_time_epoch
  gint64 sent_time_epoch
  GstClockTime duration
  GstClockTime dts
  GstClockTime pts
  gchar gst_caps_str [GST_MQTT_MAX_LEN_GST_CAPS_STR]
  }
  guint8 _reserved_hdr [GST_MQTT_LEN_MSG_HDR]
  };

8.14.1 Detailed Description

Defined a custom data type, GstMQTTMessageHdr.

GstMQTTMessageHdr contains the information needed to parse the message data at the subscriber side and is prepended to the original message data at the publisher side.

Definition at line 49 of file mqttcommon.h.

8.14.2 Member Data Documentation

8.14.2.1 "@6

```
union { ... }
```

8.14.2.2 _reserved_hdr

```
guint8 _GstMQTTMessageHdr::_reserved_hdr[GST_MQTT_LEN_MSG_HDR]
```

Definition at line 61 of file mqttcommon.h.

8.14.2.3 base_time_epoch

```
gint64 _GstMQTTMessageHdr::base_time_epoch
```

Definition at line 54 of file mqttcommon.h.

8.14.2.4 dts

```
GstClockTime _GstMQTTMessageHdr::dts
```

Definition at line 57 of file mqttcommon.h.

8.14.2.5 duration

```
GstClockTime _GstMQTTMessageHdr::duration
```

Definition at line 56 of file mqttcommon.h.

8.14.2.6 gst_caps_str

```
gchar _GstMQTTMessageHdr::gst_caps_str[GST_MQTT_MAX_LEN_GST_CAPS_STR]
```

Definition at line 59 of file mqttcommon.h.

8.14.2.7 num_mems

```
guint _GstMQTTMessageHdr::num_mems
```

Definition at line 52 of file mqttcommon.h.

8.14.2.8 pts

```
GstClockTime _GstMQTTMessageHdr::pts
```

Definition at line 58 of file mqttcommon.h.

8.14.2.9 sent_time_epoch

```
gint64 _GstMQTTMessageHdr::sent_time_epoch
```

Definition at line 55 of file mqttcommon.h.

8.14.2.10 size_mems

```
gsize _GstMQTTMessageHdr::size_mems[GST_MQTT_MAX_NUM_MEMS]
```

Definition at line 53 of file mqttcommon.h.

The documentation for this struct was generated from the following file:

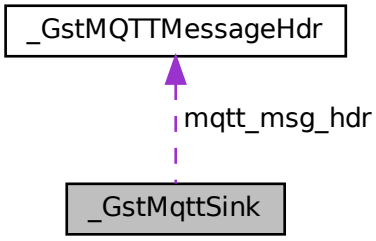
- mqtt/mqttcommon.h

8.15 _GstMqttSink Struct Reference

GstMqttSink data structure.

```
#include <mqttsink.h>
```

Collaboration diagram for _GstMqttSink:



Public Attributes

- GstBaseSink parent
- GstCaps * in_caps
- gint num_buffers
- gsize max_msg_buf_size
- GQuark gquark_err_tag
- GError * err
- gint64 base_time_epoch
- gchar * mqtt_client_id
- gchar * mqtt_host_address
- gchar * mqtt_host_port
- gchar * mqtt_topic
- gulong mqtt_pub_wait_timeout
- GMutex mqtt_sink_mutex
- GCond mqtt_sink_gcond
- mqtt_sink_state_t mqtt_sink_state
- gboolean debug

- gint mqtt_qos
- gboolean mqtt_ntp_sync
- guint mqtt_ntp_num_srvs
- gchar * mqtt_ntp_srvs
- gchar ** mqtt_ntp_hnames
- guint16 * mqtt_ntp_ports
- gboolean is_connected
- mqtt_get_unix_epoch get_epoch_func
- GstMQTTMessageHdr mqtt_msg_hdr
- gpointer mqtt_msg_buf
- gsize mqtt_msg_buf_size
- MQTTAsync mqtt_client_handle
- MQTTAsync_connectOptions mqtt_conn_opts
- MQTTAsync_responseOptions mqtt_respn_opts

8.15.1 Detailed Description

GstMqttSink data structure.

GstMqttSink inherits GstBaseSink.

Definition at line 60 of file mqttsink.h.

8.15.2 Member Data Documentation

8.15.2.1 base_time_epoch

```
gint64 _GstMqttSink::base_time_epoch
```

Definition at line 67 of file mqttsink.h.

8.15.2.2 debug

```
gboolean _GstMqttSink::debug
```

Definition at line 76 of file mqttsink.h.

8.15.2.3 err

```
GError* _GstMqttSink::err
```

Definition at line 66 of file mqttsink.h.

8.15.2.4 get_epoch_func

```
mqtt_get_unix_epoch _GstMqttSink::get_epoch_func
```

Definition at line 85 of file mqttsink.h.

8.15.2.5 gquark_err_tag

```
GQuark _GstMqttSink::gquark_err_tag
```

Definition at line 65 of file mqttsink.h.

8.15.2.6 in_caps

```
GstCaps* _GstMqttSink::in_caps
```

Definition at line 62 of file mqttsink.h.

8.15.2.7 is_connected

```
gboolean _GstMqttSink::is_connected
```

Definition at line 83 of file mqttsink.h.

8.15.2.8 max_msg_buf_size

```
gsize _GstMqttSink::max_msg_buf_size
```

Definition at line 64 of file mqttsink.h.

8.15.2.9 mqtt_client_handle

```
MQTTAsync _GstMqttSink::mqtt_client_handle
```

Definition at line 91 of file mqttsink.h.

8.15.2.10 mqtt_client_id

```
gchar* _GstMqttSink::mqtt_client_id
```

Definition at line 68 of file mqttsink.h.

8.15.2.11 mqtt_conn_opts

```
MQTTAsync_connectOptions _GstMqttSink::mqtt_conn_opts
```

Definition at line 92 of file mqttsink.h.

8.15.2.12 mqtt_host_address

```
gchar* _GstMqttSink::mqtt_host_address
```

Definition at line 69 of file mqttsink.h.

8.15.2.13 mqtt_host_port

```
gchar* _GstMqttSink::mqtt_host_port
```

Definition at line 70 of file mqttsink.h.

8.15.2.14 mqtt_msg_buf

```
gpointer _GstMqttSink::mqtt_msg_buf
```

Definition at line 88 of file mqttsink.h.

8.15.2.15 mqtt_msg_buf_size

```
gsize _GstMqttSink::mqtt_msg_buf_size
```

Definition at line 89 of file mqttsink.h.

8.15.2.16 mqtt_msg_hdr

GstMQTTMessageHdr _GstMqttSink::mqtt_msg_hdr

Definition at line 87 of file mqttsink.h.

8.15.2.17 mqtt_ntp_hnames

gchar** _GstMqttSink::mqtt_ntp_hnames

Definition at line 81 of file mqttsink.h.

8.15.2.18 mqtt_ntp_num_srvs

guint _GstMqttSink::mqtt_ntp_num_srvs

Definition at line 79 of file mqttsink.h.

8.15.2.19 mqtt_ntp_ports

guint16* _GstMqttSink::mqtt_ntp_ports

Definition at line 82 of file mqttsink.h.

8.15.2.20 mqtt_ntp_srvs

gchar* _GstMqttSink::mqtt_ntp_srvs

Definition at line 80 of file mqttsink.h.

8.15.2.21 mqtt_ntp_sync

gboolean _GstMqttSink::mqtt_ntp_sync

Definition at line 78 of file mqttsink.h.

8.15.2.22 mqtt_pub_wait_timeout

```
gulong _GstMqttSink::mqtt_pub_wait_timeout
```

Definition at line 72 of file mqttsink.h.

8.15.2.23 mqtt_qos

```
gint _GstMqttSink::mqtt_qos
```

Definition at line 77 of file mqttsink.h.

8.15.2.24 mqtt_respn_opts

```
MQTTAsync_responseOptions _GstMqttSink::mqtt_respn_opts
```

Definition at line 93 of file mqttsink.h.

8.15.2.25 mqtt_sink_gcond

```
GCond _GstMqttSink::mqtt_sink_gcond
```

Definition at line 74 of file mqttsink.h.

8.15.2.26 mqtt_sink_mutex

```
GMutex _GstMqttSink::mqtt_sink_mutex
```

Definition at line 73 of file mqttsink.h.

8.15.2.27 mqtt_sink_state

```
mqtt_sink_state_t _GstMqttSink::mqtt_sink_state
```

Definition at line 75 of file mqttsink.h.

8.15.2.28 mqtt_topic

```
gchar* _GstMqttSink::mqtt_topic
```

Definition at line 71 of file mqttsink.h.

8.15.2.29 num_buffers

```
gint _GstMqttSink::num_buffers
```

Definition at line 63 of file mqttsink.h.

8.15.2.30 parent

```
GstBaseSink _GstMqttSink::parent
```

Definition at line 61 of file mqttsink.h.

The documentation for this struct was generated from the following file:

- mqtt/mqttsink.h

8.16 _GstMqttSinkClass Struct Reference

GstMqttSinkClass data structure.

```
#include <mqttsink.h>
```

Public Attributes

- GstBaseSinkClass parent_class

8.16.1 Detailed Description

GstMqttSinkClass data structure.

GstMqttSinkClass inherits GstBaseSinkClass.

Definition at line 101 of file mqttsink.h.

8.16.2 Member Data Documentation

8.16.2.1 parent_class

```
GstBaseSinkClass _GstMqttSinkClass::parent_class
```

Definition at line 102 of file mqttsink.h.

The documentation for this struct was generated from the following file:

- mqtt/mqttsink.h

8.17 _GstMqttSrc Struct Reference

GstMqttSrc data structure.

```
#include <mqttsrc.h>
```

Public Attributes

- GstBaseSrc parent
- GstCaps * caps
- GQuark gquark_err_tag
- GError * err
- gint64 base_time_epoch
- GstClockTime latency
- gchar * mqtt_client_id
- gchar * mqtt_host_address
- gchar * mqtt_host_port
- gchar * mqtt_topic
- gint64 mqtt_sub_timeout
- gboolean debug
- gboolean is_live
- guint64 num_dumped
- gint mqtt_qos
- GAsyncQueue * aqueue
- GMutex mqtt_src_mutex
- GCond mqtt_src_gcond
- gboolean is_connected
- gboolean is_subscribed
- MQTTAsync mqtt_client_handle
- MQTTAsync_connectOptions mqtt_conn_opts
- MQTTAsync_responseOptions mqtt_respn_opts

8.17.1 Detailed Description

GstMqttSrc data structure.

GstMqttSrc inherits GstBaseSrc.

Definition at line 46 of file mqttsrc.h.

8.17.2 Member Data Documentation

8.17.2.1 aqueue

```
GAsyncQueue* _GstMqttSrc::aqueue
```

Definition at line 63 of file mqttsrc.h.

8.17.2.2 base_time_epoch

```
gint64 _GstMqttSrc::base_time_epoch
```

Definition at line 51 of file mqttsrc.h.

8.17.2.3 caps

```
GstCaps* _GstMqttSrc::caps
```

Definition at line 48 of file mqttsrc.h.

8.17.2.4 debug

```
gboolean _GstMqttSrc::debug
```

Definition at line 58 of file mqttsrc.h.

8.17.2.5 err

```
GError* _GstMqttSrc::err
```

Definition at line 50 of file mqttsrc.h.

8.17.2.6 gquark_err_tag

```
GQuark _GstMqttSrc::gquark_err_tag
```

Definition at line 49 of file mqttsrc.h.

8.17.2.7 is_connected

```
gboolean _GstMqttSrc::is_connected
```

Definition at line 66 of file mqttsrc.h.

8.17.2.8 is_live

```
gboolean _GstMqttSrc::is_live
```

Definition at line 59 of file mqttsrc.h.

8.17.2.9 is_subscribed

```
gboolean _GstMqttSrc::is_subscribed
```

Definition at line 67 of file mqttsrc.h.

8.17.2.10 latency

```
GstClockTime _GstMqttSrc::latency
```

Definition at line 52 of file mqttsrc.h.

8.17.2.11 mqtt_client_handle

```
MQTTAsync _GstMqttSrc::mqtt_client_handle
```

Definition at line 69 of file mqttsrc.h.

8.17.2.12 mqtt_client_id

```
gchar* _GstMqttSrc::mqtt_client_id
```

Definition at line 53 of file mqttsrc.h.

8.17.2.13 mqtt_conn_opts

```
MQTTAsync_connectOptions _GstMqttSrc::mqtt_conn_opts
```

Definition at line 70 of file mqttsrc.h.

8.17.2.14 mqtt_host_address

```
gchar* _GstMqttSrc::mqtt_host_address
```

Definition at line 54 of file mqttsrc.h.

8.17.2.15 mqtt_host_port

```
gchar* _GstMqttSrc::mqtt_host_port
```

Definition at line 55 of file mqttsrc.h.

8.17.2.16 mqtt_qos

```
gint _GstMqttSrc::mqtt_qos
```

Definition at line 61 of file mqttsrc.h.

8.17.2.17 mqtt_respn_opts

```
MQTTAsync_responseOptions _GstMqttSrc::mqtt_respn_opts
```

Definition at line 71 of file mqttsrc.h.

8.17.2.18 mqtt_src_gcond

```
GCond _GstMqttSrc::mqtt_src_gcond
```

Definition at line 65 of file mqttsrc.h.

8.17.2.19 mqtt_src_mutex

```
GMutex _GstMqttSrc::mqtt_src_mutex
```

Definition at line 64 of file mqttsrc.h.

8.17.2.20 mqtt_sub_timeout

```
gint64 _GstMqttSrc::mqtt_sub_timeout
```

Definition at line 57 of file mqttsrc.h.

8.17.2.21 mqtt_topic

```
gchar* _GstMqttSrc::mqtt_topic
```

Definition at line 56 of file mqttsrc.h.

8.17.2.22 num_dumped

```
guint64 _GstMqttSrc::num_dumped
```

Definition at line 60 of file mqttsrc.h.

8.17.2.23 parent

```
GstBaseSrc _GstMqttSrc::parent
```

Definition at line 47 of file mqttsrc.h.

The documentation for this struct was generated from the following file:

- mqtt/mqttsrc.h

8.18 _GstMqttSrcClass Struct Reference

GstMqttSrcClass data structure.

```
#include <mqttsrc.h>
```

Public Attributes

- GstBaseSrcClass parent_class

8.18.1 Detailed Description

GstMqttSrcClass data structure.

GstMqttSrcClass inherits GstBaseSrcClass.

Definition at line 79 of file mqttsrc.h.

8.18.2 Member Data Documentation

8.18.2.1 parent_class

```
GstBaseSrcClass _GstMqttSrcClass::parent_class
```

Definition at line 80 of file mqttsrc.h.

The documentation for this struct was generated from the following file:

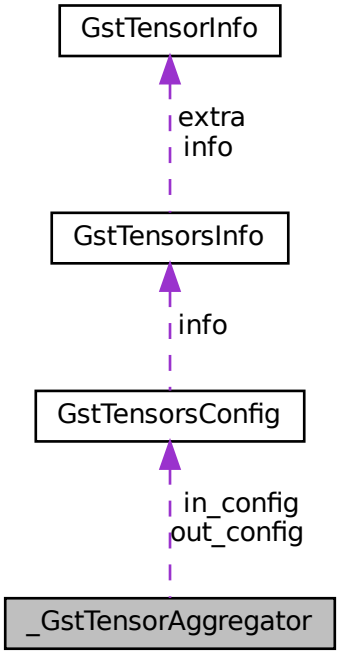
- mqtt/mqttsrc.h

8.19 _GstTensorAggregator Struct Reference

GstTensorAggregator data structure.

`#include <gsttensor_aggregator.h>`

Collaboration diagram for _GstTensorAggregator:



Public Attributes

- GstElement element
- GstPad * sinkpad
- GstPad * srcpad
- gboolean silent
- gboolean concat
- guint frames_in
- guint frames_out
- guint frames_flush
- guint frames_dim
- GHashTable * adapter_table
- gboolean tensor_configured
- GstTensorsConfig in_config
- GstTensorsConfig out_config

8.19.1 Detailed Description

GstTensorAggregator data structure.

Definition at line 52 of file gsttensor_aggregator.h.

8.19.2 Member Data Documentation

8.19.2.1 adapter_table

```
GHashTable* _GstTensorAggregator::adapter_table
```

adapt incoming tensor

Definition at line 66 of file gsttensor_aggregator.h.

8.19.2.2 concat

```
gboolean _GstTensorAggregator::concat
```

true to concatenate output buffer

Definition at line 60 of file gsttensor_aggregator.h.

8.19.2.3 element

```
GstElement _GstTensorAggregator::element
```

parent object

Definition at line 54 of file gsttensor_aggregator.h.

8.19.2.4 frames_dim

```
guint _GstTensorAggregator::frames_dim
```

index of frames in tensor dimension

Definition at line 64 of file gsttensor_aggregator.h.

8.19.2.5 frames_flush

```
guint _GstTensorAggregator::frames_flush
```

number of frames to flush

Definition at line 63 of file gsttensor_aggregator.h.

8.19.2.6 frames_in

```
guint _GstTensorAggregator::frames_in
```

number of frames in input buffer

Definition at line 61 of file gsttensor_aggregator.h.

8.19.2.7 frames_out

```
guint _GstTensorAggregator::frames_out
```

number of frames in output buffer

Definition at line 62 of file gsttensor_aggregator.h.

8.19.2.8 in_config

```
GstTensorsConfig _GstTensorAggregator::in_config
```

input tensor info

Definition at line 69 of file gsttensor_aggregator.h.

8.19.2.9 out_config

```
GstTensorsConfig _GstTensorAggregator::out_config
```

output tensor info

Definition at line 70 of file gsttensor_aggregator.h.

8.19.2.10 silent

```
gboolean _GstTensorAggregator::silent
```

true to print minimized log

Definition at line 59 of file gsttensor_aggregator.h.

8.19.2.11 sinkpad

```
GstPad* _GstTensorAggregator::sinkpad
```

sink pad

Definition at line 56 of file gsttensor_aggregator.h.

8.19.2.12 srcpad

```
GstPad* _GstTensorAggregator::srcpad
```

src pad

Definition at line 57 of file gsttensor_aggregator.h.

8.19.2.13 tensor_configured

```
gboolean _GstTensorAggregator::tensor_configured
```

True if already successfully configured tensor metadata

Definition at line 68 of file gsttensor_aggregator.h.

The documentation for this struct was generated from the following file:

- nnstreamer/elements/gsttensor_aggregator.h

8.20 _GstTensorAggregatorClass Struct Reference

GstTensorAggregatorClass data structure.

```
#include <gsttensor_aggregator.h>
```

Public Attributes

- GstElementClass parent_class

8.20.1 Detailed Description

GstTensorAggregatorClass data structure.

Definition at line 76 of file gsttensor_aggregator.h.

8.20.2 Member Data Documentation

8.20.2.1 parent_class

```
GstElementClass _GstTensorAggregatorClass::parent_class
```

parent class

Definition at line 78 of file gsttensor_aggregator.h.

The documentation for this struct was generated from the following file:

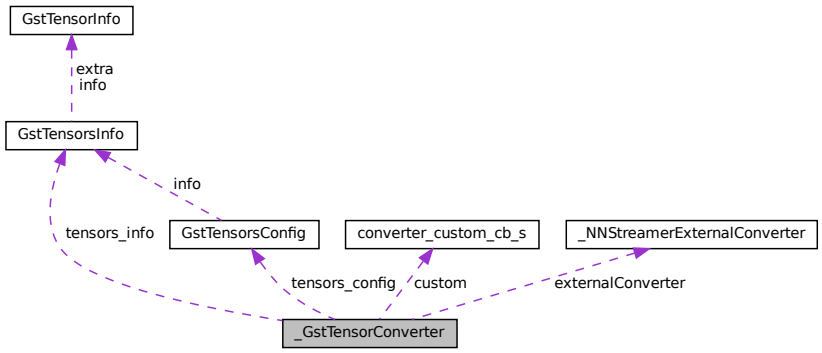
- nnstreamer/elements/gsttensor_aggregator.h

8.21 _GstTensorConverter Struct Reference

Internal data structure for tensor_converter instances.

```
#include <gsttensor_converter.h>
```

Collaboration diagram for _GstTensorConverter:



Public Attributes

- GstElement element
- GstPad * sinkpad
- GstPad * srcpad
- gboolean silent
- gboolean set_timestamp
- guint frames_per_tensor
- GstTensorsInfo tensors_info
- GHashTable * adapter_table
- media_type in_media_type
- const NNStreamerExternalConverter * externalConverter
- gsize frame_size
- gboolean remove_padding
- gboolean tensors_configured
- GstTensorsConfig tensors_config
- gboolean have_segment
- gboolean need_segment
- GstSegment segment
- GstClockTime old_timestamp
- tensor_converter_mode mode
- gchar * mode_option
- gchar * ext_fw
- converter_custom_cb_s custom
- gboolean do_not_append_header
- void * priv_data

8.21.1 Detailed Description

Internal data structure for tensor_converter instances.

Definition at line 75 of file gsttensor_converter.h.

8.21.2 Member Data Documentation

8.21.2.1 adapter_table

```
GHashTable* _GstTensorConverter::adapter_table
```

adapt incoming media stream

Definition at line 87 of file gsttensor_converter.h.

8.21.2.2 custom

converter_custom_cb_s _GstTensorConverter::custom

Definition at line 106 of file gsttensor_converter.h.

8.21.2.3 do_not_append_header

gboolean _GstTensorConverter::do_not_append_header

Definition at line 107 of file gsttensor_converter.h.

8.21.2.4 element

GstElement _GstTensorConverter::element

parent object

Definition at line 77 of file gsttensor_converter.h.

8.21.2.5 ext_fw

gchar* _GstTensorConverter::ext_fw

tensor converter custom mode framework

Definition at line 105 of file gsttensor_converter.h.

8.21.2.6 externalConverter

const NNStreamerExternalConverter* _GstTensorConverter::externalConverter

ExternalConverter is used if in_media_type == _NNS_MEDIA_PLUGINS

Definition at line 91 of file gsttensor_converter.h.

8.21.2.7 frame_size

```
gsize _GstTensorConverter::frame_size
```

size of one frame

Definition at line 93 of file gsttensor_converter.h.

8.21.2.8 frames_per_tensor

```
guint _GstTensorConverter::frames_per_tensor
```

number of frames in output tensor

Definition at line 84 of file gsttensor_converter.h.

8.21.2.9 have_segment

```
gboolean _GstTensorConverter::have_segment
```

True if received segment

Definition at line 98 of file gsttensor_converter.h.

8.21.2.10 in_media_type

```
media_type _GstTensorConverter::in_media_type
```

incoming media type

Definition at line 89 of file gsttensor_converter.h.

8.21.2.11 mode

```
tensor_converter_mode _GstTensorConverter::mode
```

tensor converter operating mode

Definition at line 103 of file gsttensor_converter.h.

8.21.2.12 mode_option

```
gchar* _GstTensorConverter::mode_option
```

tensor converter mode option

Definition at line 104 of file gsttensor_converter.h.

8.21.2.13 need_segment

```
gboolean _GstTensorConverter::need_segment
```

True to handle seg event

Definition at line 99 of file gsttensor_converter.h.

8.21.2.14 old_timestamp

```
GstClockTime _GstTensorConverter::old_timestamp
```

timestamp at prev buffer

Definition at line 101 of file gsttensor_converter.h.

8.21.2.15 priv_data

```
void* _GstTensorConverter::priv_data
```

plugin's private data

Definition at line 109 of file gsttensor_converter.h.

8.21.2.16 remove_padding

```
gboolean _GstTensorConverter::remove_padding
```

If true, zero-padding must be removed

Definition at line 94 of file gsttensor_converter.h.

8.21.2.17 segment

```
GstSegment _GstTensorConverter::segment
```

Segment, supposed time format

Definition at line 100 of file gsttensor_converter.h.

8.21.2.18 set_timestamp

```
gboolean _GstTensorConverter::set_timestamp
```

true to set timestamp when received a buffer with invalid timestamp

Definition at line 83 of file gsttensor_converter.h.

8.21.2.19 silent

```
gboolean _GstTensorConverter::silent
```

true to print minimized log

Definition at line 82 of file gsttensor_converter.h.

8.21.2.20 sinkpad

```
GstPad* _GstTensorConverter::sinkpad
```

sink pad

Definition at line 79 of file gsttensor_converter.h.

8.21.2.21 srcpad

```
GstPad* _GstTensorConverter::srcpad
```

src pad

Definition at line 80 of file gsttensor_converter.h.

8.21.2.22 tensors_config

```
GstTensorsConfig _GstTensorConverter::tensors_config
```

output tensors info

Definition at line 96 of file gsttensor_converter.h.

8.21.2.23 tensors_configured

```
gboolean _GstTensorConverter::tensors_configured
```

True if already successfully configured tensors metadata

Definition at line 95 of file gsttensor_converter.h.

8.21.2.24 tensors_info

```
GstTensorsInfo _GstTensorConverter::tensors_info
```

data structure to get/set tensor info

Definition at line 85 of file gsttensor_converter.h.

The documentation for this struct was generated from the following file:

- nnstreamer/elements/gsttensor_converter.h

8.22 _GstTensorConverterClass Struct Reference

GstTensorConverterClass data structure.

```
#include <gsttensor_converter.h>
```

Public Attributes

- GstElementClass parent_class

8.22.1 Detailed Description

GstTensorConverterClass data structure.

Definition at line 115 of file gsttensor_converter.h.

8.22.2 Member Data Documentation

8.22.2.1 parent_class

```
GstElementClass _GstTensorConverterClass::parent_class
```

parent class

Definition at line 117 of file gsttensor_converter.h.

The documentation for this struct was generated from the following file:

- nnstreamer/elements/gsttensor_converter.h

8.23 _GstTensorCrop Struct Reference

GstTensorCrop data structure.

```
#include <gsttensor_crop.h>
```

Public Attributes

- GstElement element
- GstPad * sinkpad_raw
- GstPad * sinkpad_info
- GstPad * srcpad
- gint lateness
- gboolean silent
- gboolean send_stream_start
- GstCollectPads * collect

8.23.1 Detailed Description

GstTensorCrop data structure.

Definition at line 49 of file gsttensor_crop.h.

8.23.2 Member Data Documentation

8.23.2.1 collect

```
GstCollectPads* _GstTensorCrop::collect
```

sink pads

Definition at line 61 of file gsttensor_crop.h.

8.23.2.2 element

```
GstElement _GstTensorCrop::element
```

parent object

Definition at line 51 of file gsttensor_crop.h.

8.23.2.3 lateness

```
gint _GstTensorCrop::lateness
```

time-diff of raw and info buffer

Definition at line 58 of file gsttensor_crop.h.

8.23.2.4 send_stream_start

```
gboolean _GstTensorCrop::send_stream_start
```

flag to send STREAM_START event

Definition at line 60 of file gsttensor_crop.h.

8.23.2.5 silent

```
gboolean _GstTensorCrop::silent
```

true to print minimized log

Definition at line 59 of file gsttensor_crop.h.

8.23.2.6 sinkpad_info

```
GstPad* _GstTensorCrop::sinkpad_info
```

sink pad (crop info)

Definition at line 54 of file gsttensor_crop.h.

8.23.2.7 sinkpad_raw

```
GstPad* _GstTensorCrop::sinkpad_raw
```

sink pad (raw data)

Definition at line 53 of file gsttensor_crop.h.

8.23.2.8 srcpad

```
GstPad* _GstTensorCrop::srcpad
```

src pad

Definition at line 55 of file gsttensor_crop.h.

The documentation for this struct was generated from the following file:

- nnstreamer/elements/gsttensor_crop.h

8.24 _GstTensorCropClass Struct Reference

GstTensorCropClass data structure.

```
#include <gsttensor_crop.h>
```

Public Attributes

- GstElementClass parent_class

8.24.1 Detailed Description

GstTensorCropClass data structure.

Definition at line 67 of file gsttensor_crop.h.

8.24.2 Member Data Documentation

8.24.2.1 parent_class

```
GstElementClass _GstTensorCropClass::parent_class
```

parent class

Definition at line 69 of file gsttensor_crop.h.

The documentation for this struct was generated from the following file:

- nnstreamer/elements/gsttensor_crop.h

8.25 _GstTensorDebug Struct Reference

Internal data structure for tensor_debug instances.

```
#include <gsttensor_debug.h>
```

Public Attributes

- GstBaseTransform element
- GstPad * sinkpad
- GstPad * srcpad
- gboolean silent
- tdbg_output_mode output_mode
- tdbg_cap_mode cap_mode
- tdbg_meta_mode meta_mode

8.25.1 Detailed Description

Internal data structure for tensor_debug instances.

Definition at line 84 of file gsttensor_debug.h.

8.25.2 Member Data Documentation

8.25.2.1 cap_mode

```
tdbg_cap_mode _GstTensorDebug::cap_mode
```

Definition at line 94 of file gsttensor_debug.h.

8.25.2.2 element

```
GstBaseTransform _GstTensorDebug::element
```

This is the parent object

Definition at line 86 of file gsttensor_debug.h.

8.25.2.3 meta_mode

```
tdbg_meta_mode _GstTensorDebug::meta_mode
```

Definition at line 95 of file gsttensor_debug.h.

8.25.2.4 output_mode

```
tdbg_output_mode _GstTensorDebug::output_mode
```

Definition at line 93 of file gsttensor_debug.h.

8.25.2.5 silent

```
gboolean _GstTensorDebug::silent
```

true to print minimized log

Definition at line 91 of file gsttensor_debug.h.

8.25.2.6 sinkpad

```
GstPad* _GstTensorDebug::sinkpad
```

sink pad

Definition at line 88 of file gsttensor_debug.h.

8.25.2.7 srcpad

```
GstPad* _GstTensorDebug::srcpad
```

src pad

Definition at line 89 of file gsttensor_debug.h.

The documentation for this struct was generated from the following file:

- nnstreamer/elements/gsttensor_debug.h

8.26 _GstTensorDebugClass Struct Reference

GstTensorDebugClass data structure.

```
#include <gsttensor_debug.h>
```

Public Attributes

- GstBaseTransformClass parent_class

8.26.1 Detailed Description

GstTensorDebugClass data structure.

Definition at line 101 of file gsttensor_debug.h.

8.26.2 Member Data Documentation

8.26.2.1 parent_class

```
GstBaseTransformClass _GstTensorDebugClass::parent_class
```

parent class = transform

Definition at line 103 of file gsttensor_debug.h.

The documentation for this struct was generated from the following file:

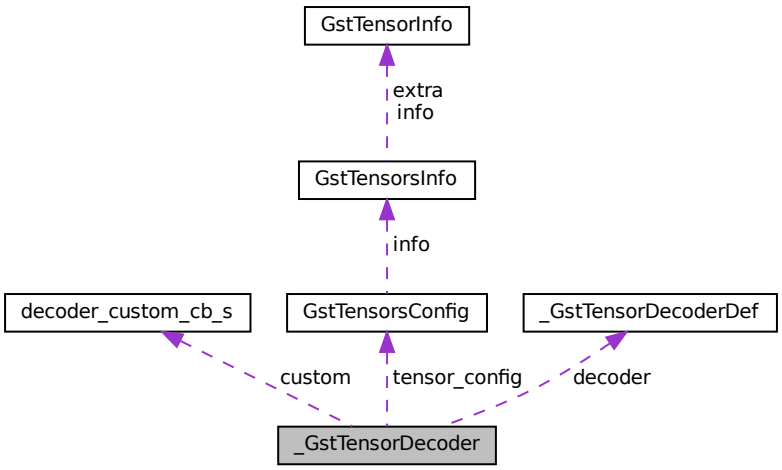
- nnstreamer/elements/gsttensor_debug.h

8.27 _GstTensorDecoder Struct Reference

Internal data structure for tensordec instances.

`#include <gsttensor_decoder.h>`

Collaboration diagram for _GstTensorDecoder:



Public Attributes

- GstBaseTransform element
- gboolean negotiated
- gboolean silent
- gchar * config_path
- gchar * option [TensorDecMaxOpNum]
- gboolean configured
- GstTensorsConfig tensor_config
- gboolean is_custom
- decoder_custom_cb_s custom
- const GstTensorDecoderDef * decoder
- void * plugin_data

8.27.1 Detailed Description

Internal data structure for tensordec instances.

Definition at line 67 of file gsttensor_decoder.h.

8.27.2 Member Data Documentation

8.27.2.1 config_path

```
gchar* _GstTensorDecoder::config_path
```

Path to configuration file

Definition at line 74 of file gsttensor_decoder.h.

8.27.2.2 configured

```
gboolean _GstTensorDecoder::configured
```

For Tensor TRUE if already successfully configured tensor metadata

Definition at line 78 of file gsttensor_decoder.h.

8.27.2.3 custom

```
decoder_custom_cb_s _GstTensorDecoder::custom
```

Definition at line 83 of file gsttensor_decoder.h.

8.27.2.4 decoder

```
const GstTensorDecoderDef* _GstTensorDecoder::decoder
```

Plugin object

Definition at line 85 of file gsttensor_decoder.h.

8.27.2.5 element

```
GstBaseTransform _GstTensorDecoder::element
```

This is the parent object

Definition at line 69 of file gsttensor_decoder.h.

8.27.2.6 is_custom

```
gboolean _GstTensorDecoder::is_custom
```

For tensor decoder custom

Definition at line 82 of file gsttensor_decoder.h.

8.27.2.7 negotiated

```
gboolean _GstTensorDecoder::negotiated
```

For transformer TRUE if tensor metadata is set

Definition at line 72 of file gsttensor_decoder.h.

8.27.2.8 option

```
gchar* _GstTensorDecoder::option[TensorDecMaxOpNum]
```

Assume we have two options

Definition at line 75 of file gsttensor_decoder.h.

8.27.2.9 plugin_data

```
void* _GstTensorDecoder::plugin_data
```

Definition at line 86 of file gsttensor_decoder.h.

8.27.2.10 silent

```
gboolean _GstTensorDecoder::silent
```

True if logging is minimized

Definition at line 73 of file gsttensor_decoder.h.

8.27.2.11 tensor_config

```
GstTensorsConfig _GstTensorDecoder::tensor_config
```

configured tensor info

Todo support tensors in the future

Definition at line 79 of file gsttensor_decoder.h.

The documentation for this struct was generated from the following file:

- nnstreamer/elements/gsttensor_decoder.h

8.28 _GstTensorDecoderClass Struct Reference

GstTensorDecoderClass inherits GstBaseTransformClass.

```
#include <gsttensor_decoder.h>
```

Public Attributes

- GstBaseTransformClass parent_class

8.28.1 Detailed Description

GstTensorDecoderClass inherits GstBaseTransformClass.

Referring another child (sibling), GstVideoFilter (abstract class) and its child (concrete class) GstVideoConverter. Note that GstTensorDecoderClass is a concrete class; thus we need to look at both.

Definition at line 96 of file gsttensor_decoder.h.

8.28.2 Member Data Documentation

8.28.2.1 parent_class

```
GstBaseTransformClass _GstTensorDecoderClass::parent_class
```

Inherits GstBaseTransformClass

Definition at line 98 of file gsttensor_decoder.h.

The documentation for this struct was generated from the following file:

- nnstreamer/elements/gsttensor_decoder.h

8.29 _GstTensorDecoderDef Struct Reference

Decoder definitions for different semantics of tensors This allows developers to create their own decoders.

`#include <nnstreamer_plugin_api_decoder.h>`

Public Attributes

- char * modename
- int(* init)(void **private_data)
- void(* exit)(void **private_data)
- int(* setOption)(void **private_data, int opNum, const char *param)
- GstCaps *(* getOutCaps)(void **private_data, const GstTensorsConfig *config)
- GstFlowReturn(* decode)(void **private_data, const GstTensorsConfig *config, const GstTensorMemory *input, GstBuffer *outbuf)
- size_t(* getTransformSize)(void **private_data, const GstTensorsConfig *config, GstCaps *caps, size_t size, GstCaps *othercaps, GstPadDirection direction)

8.29.1 Detailed Description

Decoder definitions for different semantics of tensors This allows developers to create their own decoders.

Definition at line 38 of file nnstreamer_plugin_api_decoder.h.

8.29.2 Member Data Documentation

8.29.2.1 decode

```
GstFlowReturn(* _GstTensorDecoderDef::decode) (void **private_data, const GstTensorsConfig
*config, const GstTensorMemory *input, GstBuffer *outbuf)
```

The function to be called when the input tensor incomes into tensor_decoder. The sub-plugin should update the output buffer. outbuf must be allocated but empty (gst_buffer_get_size (outbuf) == 0).

Parameters

	[in/out]	private_data A sub-plugin may save its internal private data here. The sub-plugin is responsible for alloc/free of this pointer.
in	*config*	The structure of input tensor info.
in	*input*	The array of input tensor data. The maximum array size of input data is NNS_TENSOR_SIZE_LIMIT.
out	*outbuf*	A sub-plugin should update or append proper memory for the negotiated media type.

Returns

GST_FLOW_OK if OK.

Definition at line 71 of file nnstreamer_plugin_api_decoder.h.

8.29.2.2 exit

```
void(* _GstTensorDecoderDef::exit) (void **private_data)
```

Object destruction for the decoder.

Parameters

[in/out]	private_data A sub-plugin may save its internal private data here. The sub-plugin is responsible for alloc/free of this pointer. Normally, the sub-plugin may free the private_data with this function.

Definition at line 48 of file nnstreamer_plugin_api_decoder.h.

8.29.2.3 getOutCaps

```
GstCaps*(* _GstTensorDecoderDef::getOutCaps) (void **private_data, const GstTensorsConfig *config)
```

The caller should unref the returned GstCaps using gst_caps_unref(). The sub-plugin should validate the information of input tensor and return proper media type. Note that the information of input tensor is not a fixed value and the pipeline may try different values during the cap negotiations. Do NOT allocate or fix internal data structure until decode is called.

Parameters

	[in/out]	private_data A sub-plugin may save its internal private data here. The sub-plugin is responsible for alloc/free of this pointer.
in	*config*	The structure of input tensor info.

Returns

GstCaps object describing media type.

Definition at line 61 of file nnstreamer_plugin_api_decoder.h.

8.29.2.4 getTransformSize

```
size_t(* _GstTensorDecoderDef::getTransformSize) (void **private_data, const GstTensorsConfig
*config, GstCaps *caps, size_t size, GstCaps *othercaps, GstPadDirection direction)
```

Optional. The sub-plugin may calculate the size in bytes of a buffer. If this is NULL, tensor_decoder will pass the empty buffer and the sub-plugin should append the memory block when called decode. See GstBaseTransform↩ Class::transform_size for the details.

Parameters

	[in/out]	private_data A sub-plugin may save its internal private data here. The sub-plugin is responsible for alloc/free of this pointer.
in	*config*	The structure of input tensor info.
in	*caps*	GstCaps object for the given direction.
in	*size*	The size of a buffer for the given direction.
in	*othercaps*	GstCaps object on the other pad for the given direction.
in	*direction*	The direction of a pad. Normally this is GST_PAD_SINK.

Returns

The size of a buffer.

Definition at line 82 of file nnstreamer_plugin_api_decoder.h.

8.29.2.5 init

```
int(* _GstTensorDecoderDef::init) (void **private_data)
```

Object initialization for the decoder.

Parameters

[in/out]	private_data A sub-plugin may save its internal private data here. The sub-plugin is responsible for alloc/free of this pointer. Normally, the sub-plugin may allocate the private_data with this function.

Returns

TRUE if OK. FALSE if error.

Definition at line 42 of file nnstreamer_plugin_api_decoder.h.

8.29.2.6 modename

```
char* _GstTensorDecoderDef::modename
```

Unique decoder name. GST users choose decoders with mode="modename".

Definition at line 40 of file nnstreamer_plugin_api_decoder.h.

8.29.2.7 setOption

```
int(* _GstTensorDecoderDef::setOption) (void **private_data, int opNum, const char *param)
```

Process with the given options. It can be called repeatedly.

Parameters

	[in/out]	private_data A sub-plugin may save its internal private data here. The sub-plugin is responsible for alloc/free of this pointer.
in	*opNum*	The index of the given options.
in	*param*	The option string. A sub-plugin should parse the string to get the proper value.

Returns

TRUE if OK. FALSE if error.

Definition at line 53 of file nnstreamer_plugin_api_decoder.h.

The documentation for this struct was generated from the following file:

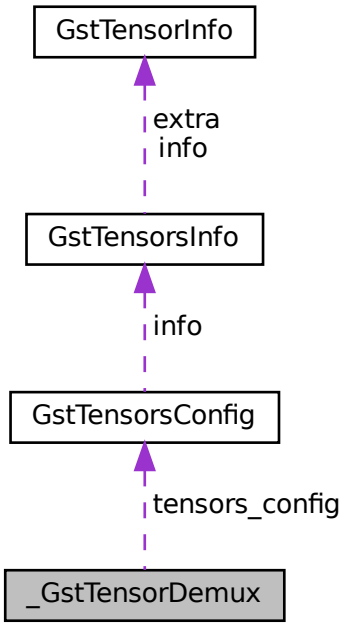
- nnstreamer/include/nnstreamer_plugin_api_decoder.h

8.30 _GstTensorDemux Struct Reference

Tensor Muxer data structure.

```
#include <gsttensor_demux.h>
```

Collaboration diagram for _GstTensorDemux:



Public Attributes

- GstElement element
- gboolean silent
- GstPad * sinkpad
- GSList * srcpads
- guint32 num_srcpads
- GList * tensorpick
- gboolean have_group_id
- guint group_id
- GstTensorsConfig tensors_config

8.30.1 Detailed Description

Tensor Muxer data structure.

Definition at line 50 of file gsttensor_demux.h.

8.30.2 Member Data Documentation

8.30.2.1 element

```
GstElement _GstTensorDemux::element
```

Definition at line 52 of file gsttensor_demux.h.

8.30.2.2 group_id

```
guint _GstTensorDemux::group_id
```

Definition at line 60 of file gsttensor_demux.h.

8.30.2.3 have_group_id

```
gboolean _GstTensorDemux::have_group_id
```

Definition at line 59 of file gsttensor_demux.h.

8.30.2.4 num_srcpads

```
guint32 _GstTensorDemux::num_srcpads
```

Definition at line 57 of file gsttensor_demux.h.

8.30.2.5 silent

```
gboolean _GstTensorDemux::silent
```

Definition at line 54 of file gsttensor_demux.h.

8.30.2.6 sinkpad

```
GstPad* _GstTensorDemux::sinkpad
```

Definition at line 55 of file gsttensor_demux.h.

8.30.2.7 srcpads

```
GSList* _GstTensorDemux::srcpads
```

Definition at line 56 of file gsttensor_demux.h.

8.30.2.8 tensorpick

```
GList* _GstTensorDemux::tensorpick
```

Definition at line 58 of file gsttensor_demux.h.

8.30.2.9 tensors_config

```
GstTensorsConfig _GstTensorDemux::tensors_config
```

input tensors info

Definition at line 62 of file gsttensor_demux.h.

The documentation for this struct was generated from the following file:

- nnstreamer/elements/gsttensor_demux.h

8.31 _GstTensorDemuxClass Struct Reference

GstTensorDeMuxClass inherits GstElementClass.

`#include <gsttensor_demux.h>`

Public Attributes

- GstElementClass parent_class

8.31.1 Detailed Description

GstTensorDeMuxClass inherits GstElementClass.

Definition at line 68 of file gsttensor_demux.h.

8.31.2 Member Data Documentation

8.31.2.1 parent_class

`GstElementClass _GstTensorDemuxClass::parent_class`

Definition at line 70 of file gsttensor_demux.h.

The documentation for this struct was generated from the following file:

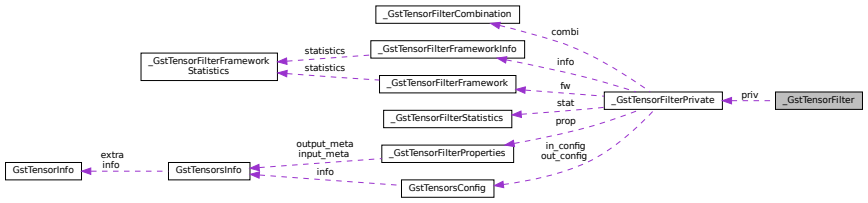
- nnstreamer/elements/gsttensor_demux.h

8.32 _GstTensorFilter Struct Reference

Internal data structure for tensor_filter instances.

`#include <tensor_filter.h>`

Collaboration diagram for _GstTensorFilter:



Public Attributes

- GstBaseTransform element
- GstTensorFilterPrivate priv
- GstClockTime prev_ts
- GstClockTimeDiff throttling_delay
- GstClockTimeDiff throttling_accum

8.32.1 Detailed Description

Internal data structure for tensor_filter instances.

Definition at line 61 of file tensor_filter.h.

8.32.2 Member Data Documentation

8.32.2.1 element

```
GstBaseTransform _GstTensorFilter::element
```

This is the parent object

Definition at line 63 of file tensor_filter.h.

8.32.2.2 prev_ts

```
GstClockTime _GstTensorFilter::prev_ts
```

previous timestamp

Definition at line 67 of file tensor_filter.h.

8.32.2.3 priv

```
GstTensorFilterPrivate _GstTensorFilter::priv
```

Internal properties for tensor-filter

Definition at line 65 of file tensor_filter.h.

8.32.2.4 throttling_accum

```
GstClockTimeDiff _GstTensorFilter::throttling_accum
```

accumulated frame durations for throttling

Definition at line 69 of file tensor_filter.h.

8.32.2.5 throttling_delay

```
GstClockTimeDiff _GstTensorFilter::throttling_delay
```

throttling delay from tensor rate

Definition at line 68 of file tensor_filter.h.

The documentation for this struct was generated from the following file:

- nnstreamer/tensor_filter/tensor_filter.h

8.33 _GstTensorFilterClass Struct Reference

GstTensorFilterClass inherits GstBaseTransformClass.

```
#include <tensor_filter.h>
```

Public Attributes

- GstBaseTransformClass parent_class

8.33.1 Detailed Description

GstTensorFilterClass inherits GstBaseTransformClass.

Referring another child (sibling), GstVideoFilter (abstract class) and its child (concrete class) GstVideoConverter. Note that GstTensorFilterClass is a concrete class; thus we need to look at both.

Definition at line 79 of file tensor_filter.h.

8.33.2 Member Data Documentation

8.33.2.1 parent_class

```
GstBaseTransformClass _GstTensorFilterClass::parent_class
```

Inherits GstBaseTransformClass

Definition at line 81 of file tensor_filter.h.

The documentation for this struct was generated from the following file:

- nnstreamer/tensor_filter/tensor_filter.h

8.34 _GstTensorFilterCombination Struct Reference

Structure definition for tensor-filter in/out combination.

```
#include <tensor_filter_common.h>
```

Public Attributes

- GList * in_combi
- GList * out_combi_i
- GList * out_combi_o
- gboolean in_combi_defined
- gboolean out_combi_i_defined
- gboolean out_combi_o_defined

8.34.1 Detailed Description

Structure definition for tensor-filter in/out combination.

Definition at line 131 of file tensor_filter_common.h.

8.34.2 Member Data Documentation

8.34.2.1 in_combi

```
GList* _GstTensorFilterCombination::in_combi
```

Select the input tensor(s) to invoke the models

Definition at line 133 of file tensor_filter_common.h.

8.34.2.2 in_combi_defined

```
gboolean _GstTensorFilterCombination::in_combi_defined
```

True if input combination is defined

Definition at line 136 of file tensor_filter_common.h.

8.34.2.3 out_combi_i

```
GList* _GstTensorFilterCombination::out_combi_i
```

Select the output tensor(s) from the input tensor(s)

Definition at line 134 of file tensor_filter_common.h.

8.34.2.4 out_combi_i_defined

```
gboolean _GstTensorFilterCombination::out_combi_i_defined
```

True if output combination from input is defined

Definition at line 137 of file tensor_filter_common.h.

8.34.2.5 out_combi_o

```
GList* _GstTensorFilterCombination::out_combi_o
```

Select the output tensor(s) from the model output

Definition at line 135 of file tensor_filter_common.h.

8.34.2.6 out_combi_o_defined

```
gboolean _GstTensorFilterCombination::out_combi_o_defined
```

True if output combination from model output is defined

Definition at line 138 of file tensor_filter_common.h.

The documentation for this struct was generated from the following file:

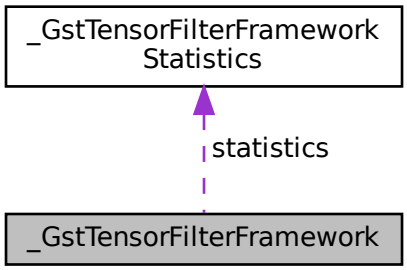
- nnstreamer/tensor_filter/tensor_filter_common.h

8.35 _GstTensorFilterFramework Struct Reference

Tensor_Filter Subplugin definition.

```
#include <nnstreamer_plugin_api_filter.h>
```

Collaboration diagram for _GstTensorFilterFramework:



Public Attributes

- uint64_t version
- int(* open)(const GstTensorFilterProperties *prop, void **private_data)
- void(* close)(const GstTensorFilterProperties *prop, void **private_data)
- union {
  struct {
  char * name
  int allow_in_place
  int allocate_in_invoke
  int run_without_model
  int verify_model_path
  const GstTensorFilterFrameworkStatistics * statistics
  int(* invoke_NN)(const
  GstTensorFilterProperties
  *prop, void **private_data,
  const GstTensorMemory *input,
  GstTensorMemory *output)
  int(* getInputDimension)(const
  GstTensorFilterProperties
  *prop, void **private_data,
  GstTensorsInfo *info)
  int(* getOutputDimension)(const
  GstTensorFilterProperties
  *prop, void **private_data,
  GstTensorsInfo *info)
  int(* setInputDimension)(const
  GstTensorFilterProperties
  *prop, void **private_data,
  const GstTensorsInfo *in_info,
  GstTensorsInfo *out_info)

```
    void(* destroyNotify )(void **private_data, void *data)
    int(* reloadModel )(const
      GstTensorFilterProperties
      *prop, void **private_data)
    int(* handleEvent )(event_ops ops,
      GstTensorFilterFrameworkEventData
      *data)
    int(* checkAvailability )(accl_hw hw)
    int(* allocateInInvoke )(void **private_data)
  }
      Tensor_Filter Subplugin definition Version 0.
  struct {
    int(* invoke )(const
      GstTensorFilterFramework *self,
      GstTensorFilterProperties
      *prop, void *private_data,
      const GstTensorMemory *input,
      GstTensorMemory *output)
    int(* getFrameworkInfo )(const
      GstTensorFilterFramework *self,
      const
      GstTensorFilterProperties
      *prop, void *private_data,
      GstTensorFilterFrameworkInfo
      *fw_info)
    int(* getModelInfo )(const
      GstTensorFilterFramework *self,
      const
      GstTensorFilterProperties
      *prop, void *private_data,
      model_info_ops ops,
      GstTensorsInfo *in_info,
      GstTensorsInfo *out_info)
    int(* eventHandler )(const
      GstTensorFilterFramework *self,
      const
      GstTensorFilterProperties
      *prop, void *private_data,
      event_ops ops,
      GstTensorFilterFrameworkEventData
      *data)
    void * subplugin_data
  }
      Tensor_Filter Subplugin definition Version 1.
};
```

Distinct elements between two versions of the subplugin interfaces.

8.35.1 Detailed Description

Tensor_Filter Subplugin definition.

Common callback parameters: prop Filter properties. Read Only. private_data Subplugin's private data. Set this (*private_data = XXX) if you want to change filter->private_data.

Definition at line 249 of file nnstreamer_plugin_api_filter.h.

8.35.2 Member Data Documentation

8.35.2.1 "@53

```
union { ... }
```

Distinct elements between two versions of the subplugin interfaces.

8.35.2.2 allocate_in_invoke

```
int _GstTensorFilterFramework::allocate_in_invoke
```

TRUE(nonzero) if invoke_NN is going to allocate output ptr by itself and return the address via output ptr. Do not change this value after cap negotiation is complete (or the stream has been started).

Definition at line 286 of file nnstreamer_plugin_api_filter.h.

8.35.2.3 allocateInInvoke

```
int(* _GstTensorFilterFramework::allocateInInvoke) (void **private_data)
```

Optional. Tensor-filter will call it when allocate_in_invoke is set to TRUE. This check if the provided model for the framework supports allocation at invoke or not. If this is not defined, then the value of allocate_in_invoke is assumed to be final for all models.

Parameters

in	*private_data*	A subplugin may save its internal private data here.

Returns

0 if supported. -errno if not supported.

Definition at line 378 of file nnstreamer_plugin_api_filter.h.

8.35.2.4 allow_in_place

```
int _GstTensorFilterFramework::allow_in_place
```

TRUE(nonzero) if in-place transfer of input-to-output is allowed. Not supported in main, yet

Definition at line 285 of file nnstreamer_plugin_api_filter.h.

8.35.2.5 checkAvailability

```
int(* _GstTensorFilterFramework::checkAvailability) (accl_hw hw)
```

Optional. Check if the provided hardware accelerator is supported. This check is static or dynamic based on framework support. Positive response of this check does not guarantee successful running of model with this accelerator. The static check can be performed without opening the framework.

Parameters

in	*hw*	backend accelerator hardware

Returns

0 if supported. -errno if not supported.

Definition at line 371 of file nnstreamer_plugin_api_filter.h.

8.35.2.6 close

```
void(* _GstTensorFilterFramework::close) (const GstTensorFilterProperties *prop, void **private↩
_data)
```

Optional. Tensor-filter will not call other callbacks after calling close. Free-ing private_data is this function's responsibility. Set NULL after that.

Parameters

in	*prop*	read-only property values.
	[in/out]	private_data A subplugin may save its internal private data here. The subplugin is responsible for alloc/free of this pointer. Normally, close() frees private_data and set NULL.

Definition at line 267 of file nnstreamer_plugin_api_filter.h.

8.35.2.7 destroyNotify

```
void(* _GstTensorFilterFramework::destroyNotify) (void **private_data, void *data)
```

Optional. Tensor-filter will call it when 'allocate_in_invoke' flag of the framework is TRUE and allocateInInvoke also return enabled. Basically, it is called when the data element is destroyed. If it's set as NULL, g_free() will be used as a default. It will be helpful when the data pointer is included as an object of a nnfw. For instance, if the data pointer is removed when the object is gone, it occurs error. In this case, the objects should be maintained for a while first and destroyed when the data pointer is destroyed. Those kinds of logic could be defined at this method.

Parameters

	[in/out]	private_data A subplugin may save its internal private data here. The subplugin is responsible for alloc/free of this pointer.
in	*data*	the data element.

Definition at line 346 of file nnstreamer_plugin_api_filter.h.

8.35.2.8 eventHandler

```
int(* _GstTensorFilterFramework::eventHandler) (const GstTensorFilterFramework *self, const
GstTensorFilterProperties *prop, void *private_data, event_ops ops, GstTensorFilterFrameworkEventData
*data)
```

Mandatory callback. Runs the event corresponding to the passed operation. If ops == DESTROY_NOTIFY: Tensor-filter will call it when 'allocate_in_invoke' property of the framework is TRUE. Basically, it is called when the data element is destroyed. If it's set as NULL, g_free() will be used as a default. It will be helpful when the data pointer is included as an object of a nnfw. For instance, if the data pointer is removed when the object is gone, it occurs error. In this case, the objects should be maintained for a while first and destroyed when the data pointer is destroyed. Those kinds of logic could be defined at this method. If ops == RELOAD_MODEL: Tensor-filter will call it when a model property is newly configured. Also, 'is-updatable' property of the framework should be TRUE. This function reloads a new model passed in as argument via data. Note that it requires extra memory size enough to temporarily hold both old and new models during this function to hide the reload overhead. If ops == CUSTOM_PROP: Tensor-filter will call to update the custom properties of the subplugin. If ops == SET_INPUT_PROP: Tensor-filter will call to update the property of the subplugin. This function will take tensor info and layout as the argument. This operation can update input tensor shape, type, name and layout. If ops == SET_OUTPUT_PROP: Tensor-filter will call to update the property of the subplugin. This function will take tensor info and layout as the argument. This operation can update output tensor shape, type, name and layout. If ops == SET_ACCELERATOR: Tensor-filter will call to update the property of the subplugin. This function will take accelerator list as the argument. This operation will update the backend to be used by the corresponding subplugin. List of operations to be supported are optional. Note: In these operations, the argument 'prop' will not contain the updated information, but will be updated after the corresponding operation is succeeded.

Parameters

in	*prop*	read-only property values
	[in/out]	private_data A subplugin may save its internal private data here. The subplugin is responsible for alloc/free of this pointer. (can be NULL)
in	*ops*	operation to be performed
	[in/out]	data event data for the supported handlers (can be NULL)

Returns

0 if OK. non-zero if error. -ENOENT if operation is not supported. -EINVAL if operation is supported but provided arguments are invalid.

Definition at line 445 of file nnstreamer_plugin_api_filter.h.

8.35.2.9 getFrameworkInfo

```
int(* _GstTensorFilterFramework::getFrameworkInfo) (const GstTensorFilterFramework *self, const
GstTensorFilterProperties *prop, void *private_data, GstTensorFilterFrameworkInfo *fw_info)
```

Mandatory callback. Get the frameworks statically determined info. Argument 'private_data' can be NULL. If provided 'private_data' is not NULL, then some info, such as 'allocate_in_invoke', can be updated based on the model being used (inferred from the 'private_data' provided). This updated info is useful for custom filter, as some custom filter's ability to support 'allocate_in_invoke' depends on the opened model.

Parameters

in	*prop*	read-only property values
in	*private_data*	A subplugin may save its internal private data here. The subplugin is responsible for alloc/free of this pointer. This parameter can be NULL.
out	*fw_info*	struct to hold frameworks info. Must be allocated by the caller (return value).

Returns

0 if OK. non-zero if error.

Note

CAUTION: private_data can be NULL if the framework is not yet opened by the caller.

Definition at line 407 of file nnstreamer_plugin_api_filter.h.

8.35.2.10 getInputDimension

```
int(* _GstTensorFilterFramework::getInputDimension) (const GstTensorFilterProperties *prop,
void **private_data, GstTensorsInfo *info)
```

Optional. Set NULL if not supported. Get dimension of input tensor If getInputDimension is NULL, setInput↩ Dimension must be defined. If getInputDimension is defined, it is recommended to define getOutputDimension.

Parameters

in	*prop*	read-only property values
	[in/out]	private_data A subplugin may save its internal private data here. The subplugin is responsible for alloc/free of this pointer.
out	*info*	structure of tensor info (return value)

Returns

0 if OK. non-zero if error.

Definition at line 303 of file nnstreamer_plugin_api_filter.h.

8.35.2.11 getModelInfo

```
int(* _GstTensorFilterFramework::getModelInfo) (const GstTensorFilterFramework *self, const
GstTensorFilterProperties *prop, void *private_data, model_info_ops ops, GstTensorsInfo *in↩
_info, GstTensorsInfo *out_info)
```

Mandatory callback. Gets the model related tensor info. If ops == GET_IN_OUT_INFO, in_info would contain the input tensor info, and out_info would contain the output tensor info. If ops == SET_INPUT_INFO, in_info would contain the provided input tensor info, and out_info would contain the updated output tensor info. At least one of SET_INPUT_INFO and GET_IN_OUT_INFO operations must be supported.

Note: Tensor_Filter::main will configure input dimension from pad-cap in run-time for the sub-plugin. Then, the sub-plugin is required to return corresponding output dimension, and will call SET_INPUT_INFO operation.

Note: With SET_INPUT_INFO operation, the caller must NOT allocate or fix internal data structure based on the return value until invoke is called. GStreamer may try different dimensions before settling down.

Parameters

in	*prop*	read-only property values
	[in/out]	private_data A subplugin may save its internal private data here. The subplugin is responsible for alloc/free of this pointer.
in	*ops*	operation to be performed
	[in/out]	in_info structure of input tensor info
out	*out_info*	structure of output tensor info (return value)

Returns

0 if OK. non-zero if error. -ENOENT is operation is not supported.

Definition at line 420 of file nnstreamer_plugin_api_filter.h.

8.35.2.12 getOutputDimension

```
int(* _GstTensorFilterFramework::getOutputDimension) (const GstTensorFilterProperties *prop,
void **private_data, GstTensorsInfo *info)
```

Optional. Set NULL if not supported. Get dimension of output tensor If getInputDimension is NULL, setInput↩ Dimension must be defined. If getInputDimension is defined, it is recommended to define getOutputDimension.

Parameters

in	*prop*	read-only property values
	[in/out]	private_data A subplugin may save its internal private data here. The subplugin is responsible for alloc/free of this pointer.
out	*info*	structure of tensor info (return value)

Returns

0 if OK. non-zero if error.

Definition at line 315 of file nnstreamer_plugin_api_filter.h.

8.35.2.13 handleEvent

```
int(* _GstTensorFilterFramework::handleEvent) (event_ops ops, GstTensorFilterFrameworkEventData
*data)
```

Optional. Runs the event corresponding to the passed operation. If ops == CHECK_HW_AVAILABILITY: tensor↩ _filter will call to check the hw availability with custom option. List of operations to be supported are optional.

Parameters

in	*ops*	operation to be performed
	[in/out]	data event data for the supported handlers (can be NULL)

Returns

0 if OK. non-zero if error. -ENOENT if operation is not supported. -EINVAL if operation is supported but provided arguments are invalid.

Definition at line 361 of file nnstreamer_plugin_api_filter.h.

8.35.2.14 invoke

```
int(* _GstTensorFilterFramework::invoke) (const GstTensorFilterFramework *self, GstTensorFilterProperties
*prop, void *private_data, const GstTensorMemory *input, GstTensorMemory *output)
```

Mandatory callback. Invoke the given network model.

Parameters

	[in/out]	prop property values. In the case of dynamic invoke, the output tensors info must be filled.
	[in/out]	private_data A subplugin may save its internal private data here. The subplugin is responsible for alloc/free of this pointer.
in	*input*	The array of input tensors. Allocated and filled by tensor_filter/main
out	*output*	The array of output tensors. Allocated by tensor_filter/main and to be filled by invoke. If allocate_in_invoke is TRUE, sub-plugin should allocate the memory block for output tensor. (data in GstTensorMemory)

Returns

0 if OK. non-zero if error.

Definition at line 395 of file nnstreamer_plugin_api_filter.h.

8.35.2.15 invoke_NN

```
int(* _GstTensorFilterFramework::invoke_NN) (const GstTensorFilterProperties *prop, void **private↩
_data, const GstTensorMemory *input, GstTensorMemory *output)
```

Mandatory callback. Invoke the given network model.

Parameters

in	*prop*	read-only property values
	[in/out]	private_data A subplugin may save its internal private data here. The subplugin is responsible for alloc/free of this pointer.
in	*input*	The array of input tensors. Allocated and filled by tensor_filter/main
out	*output*	The array of output tensors. Allocated by tensor_filter/main and to be filled by invoke_NN. If allocate_in_invoke is TRUE, sub-plugin should allocate the memory block for output tensor. (data in GstTensorMemory)

Returns

0 if OK. non-zero if error.

Definition at line 292 of file nnstreamer_plugin_api_filter.h.

8.35.2.16 name

```
char* _GstTensorFilterFramework::name
```

Name of the neural network framework, searchable by FRAMEWORK property

Definition at line 284 of file nnstreamer_plugin_api_filter.h.

8.35.2.17 open

```
int(* _GstTensorFilterFramework::open) (const GstTensorFilterProperties *prop, void **private↩
_data)
```

Optional. Tensor-filter will call this before any of other callbacks and will call once before calling close.

Note: If 'open' callback is not defined, then the private_data passed in other callbacks will be NULL.

Parameters

in	*prop*	read-only property values.
	[in/out]	private_data A subplugin may save its internal private data here. The subplugin is responsible for alloc/free of this pointer. Normally, open() allocates memory for private_data.

Returns

0 if ok. < 0 if error.

Definition at line 257 of file nnstreamer_plugin_api_filter.h.

8.35.2.18 reloadModel

```
int(* _GstTensorFilterFramework::reloadModel) (const GstTensorFilterProperties *prop, void
**private_data)
```

Optional. Tensor-filter will call it when a model property is newly configured. Also, 'is-updatable' property of the framework should be TRUE. This function reloads a new model specified in the 'prop' argument. Note that it requires extra memory size enough to temporarily hold both old and new models during this function to hide the reload overhead.

Parameters

in	*prop*	read-only property values
	[in/out]	private_data A subplugin may save its internal private data here. The subplugin is responsible for alloc/free of this pointer. Normally, close() frees private_data and set NULL.

Returns

0 if ok. < 0 if error.

Definition at line 353 of file nnstreamer_plugin_api_filter.h.

8.35.2.19 run_without_model

```
int _GstTensorFilterFramework::run_without_model
```

TRUE(nonzero) when the neural network framework does not need a model file. Tensor-filter will run invoke_NN without model.

Definition at line 287 of file nnstreamer_plugin_api_filter.h.

8.35.2.20 setInputDimension

```
int(* _GstTensorFilterFramework::setInputDimension) (const GstTensorFilterProperties *prop,
void **private_data, const GstTensorsInfo *in_info, GstTensorsInfo *out_info)
```

Optional. Set Null if not supported. Tensor_Filter::main will configure input dimension from pad-cap in run-time for the sub-plugin. Then, the sub-plugin is required to return corresponding output dimension If this is NULL, both getInput/OutputDimension must be non-NULL.

When you use this, do NOT allocate or fix internal data structure based on it until invoke is called. GStreamer may try different dimensions before settling down.

Parameters

in	*prop*	read-only property values
	[in/out]	private_data A subplugin may save its internal private data here. The subplugin is responsible for alloc/free of this pointer.
in	*in_info*	structure of input tensor info
out	*out_info*	structure of output tensor info (return value)

Returns

0 if OK. non-zero if error.

Definition at line 327 of file nnstreamer_plugin_api_filter.h.

8.35.2.21 statistics

```
const GstTensorFilterFrameworkStatistics* _GstTensorFilterFramework::statistics
```

usage statistics by the framework. This is shared across all opened instances of this framework.

Definition at line 290 of file nnstreamer_plugin_api_filter.h.

8.35.2.22 subplugin_data

```
void* _GstTensorFilterFramework::subplugin_data
```

This is used by tensor_filter infrastructure. Subplugin authors should NEVER update this. Only the files in /gst/nnstreamer/tensor_filter/ are allowed to access this.

Definition at line 464 of file nnstreamer_plugin_api_filter.h.

8.35.2.23 verify_model_path

```
int _GstTensorFilterFramework::verify_model_path
```

TRUE(nonzero) when the NNS framework, not the sub-plugin, should verify the path of model files.

Definition at line 288 of file nnstreamer_plugin_api_filter.h.

8.35.2.24 version

```
uint64_t _GstTensorFilterFramework::version
```

Version of the struct | 32bit (validity check) | 16bit (API version) | 16bit (Subplugin's internal version. Tensor-filter does not care.) | API version will be 0x0 (earlier version (_GstTensorFilterFramework_v0)) or 0x1 (newer version (_GstTensorFilterFramework_v1))

Definition at line 251 of file nnstreamer_plugin_api_filter.h.

The documentation for this struct was generated from the following file:

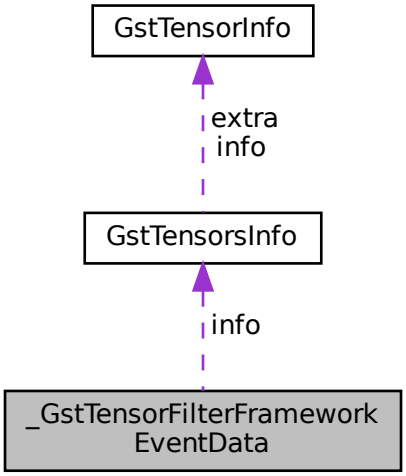
- nnstreamer/include/nnstreamer_plugin_api_filter.h

8.36 _GstTensorFilterFrameworkEventData Struct Reference

User data for the tensor_tilter subplugin related events.

```
#include <nnstreamer_plugin_api_filter.h>
```

Collaboration diagram for _GstTensorFilterFrameworkEventData:



Public Attributes

- union {
  struct {
  void * data
  }
  struct {
  const char ** model_files
  int num_models
  }
  struct {
  const char * custom_properties
  }
  struct {
  const GstTensorsInfo * info
  tensors_layout layout
  }
  struct {
  accl_hw * hw_list
  int num_hw
  }
  struct {
  accl_hw hw
  const char * custom
  }
  };

8.36.1 Detailed Description

User data for the tensor_tilter subplugin related events.

Definition at line 200 of file nnstreamer_plugin_api_filter.h.

8.36.2 Member Data Documentation

8.36.2.1 "@39

```
union { ... }
```

Union of the user data for each event supported by eventHandler

8.36.2.2 custom

```
const char* _GstTensorFilterFrameworkEventData::custom
```

custom option for hardware detection

Definition at line 235 of file nnstreamer_plugin_api_filter.h.

8.36.2.3 custom_properties

```
const char* _GstTensorFilterFrameworkEventData::custom_properties
```

sub-plugin specific custom property values in string

Definition at line 217 of file nnstreamer_plugin_api_filter.h.

8.36.2.4 data

```
void* _GstTensorFilterFrameworkEventData::data
```

The data element to be destroyed

Definition at line 206 of file nnstreamer_plugin_api_filter.h.

8.36.2.5 hw

```
accl_hw _GstTensorFilterFrameworkEventData::hw
```

accelerator to check availability

Definition at line 234 of file nnstreamer_plugin_api_filter.h.

8.36.2.6 hw_list

```
accl_hw* _GstTensorFilterFrameworkEventData::hw_list
```

accelerators supported by framework intersected with the new user provided accelerator preference

Definition at line 228 of file nnstreamer_plugin_api_filter.h.

8.36.2.7 info

```
const GstTensorsInfo* _GstTensorFilterFrameworkEventData::info
```

The tensor info to be updated to

Definition at line 222 of file nnstreamer_plugin_api_filter.h.

8.36.2.8 layout

```
tensors_layout _GstTensorFilterFrameworkEventData::layout
```

The layout of the tensor to be updated to

Definition at line 223 of file nnstreamer_plugin_api_filter.h.

8.36.2.9 model_files

```
const char** _GstTensorFilterFrameworkEventData::model_files
```

File path to the new list of model files

Definition at line 211 of file nnstreamer_plugin_api_filter.h.

8.36.2.10 num_hw

```
int _GstTensorFilterFrameworkEventData::num_hw
```

number of hardware accelerators in the hw_list supported by the framework

Definition at line 229 of file nnstreamer_plugin_api_filter.h.

8.36.2.11 num_models

```
int _GstTensorFilterFrameworkEventData::num_models
```

Updated number of the model files

Definition at line 212 of file nnstreamer_plugin_api_filter.h.

The documentation for this struct was generated from the following file:

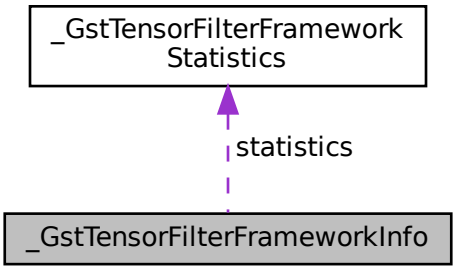
- nnstreamer/include/nnstreamer_plugin_api_filter.h

8.37 _GstTensorFilterFrameworkInfo Struct Reference

Tensor_Filter Subplugin framework related information.

`#include <nnstreamer_plugin_api_filter.h>`

Collaboration diagram for _GstTensorFilterFrameworkInfo:



Public Attributes

- const char * name
- int allow_in_place
- int allocate_in_invoke
- int run_without_model
- int verify_model_path
- const accl_hw * hw_list
- int num_hw
- accl_hw accl_auto
- accl_hw accl_default
- const GstTensorFilterFrameworkStatistics * statistics

8.37.1 Detailed Description

Tensor_Filter Subplugin framework related information.

All the information except the supported accelerator is provided statically. Accelerators can be provided based on static or dynamic check dependent on framework support.

Definition at line 158 of file nnstreamer_plugin_api_filter.h.

8.37.2 Member Data Documentation

8.37.2.1 accl_auto

```
accl_hw _GstTensorFilterFrameworkInfo::accl_auto
```

accelerator to be used in auto mode (acceleration to be used but accelerator is not specified for the filter) - default -1 implies use first entry from hw_list.

Definition at line 167 of file nnstreamer_plugin_api_filter.h.

8.37.2.2 accl_default

```
accl_hw _GstTensorFilterFrameworkInfo::accl_default
```

accelerator to be used by default (valid user input is not provided) - default -1 implies use first entry from hw_list.

Definition at line 168 of file nnstreamer_plugin_api_filter.h.

8.37.2.3 allocate_in_invoke

```
int _GstTensorFilterFrameworkInfo::allocate_in_invoke
```

TRUE(nonzero) if invoke_NN is going to allocate output ptr by itself and return the address via output ptr. Do not change this value after cap negotiation is complete (or the stream has been started).

Definition at line 162 of file nnstreamer_plugin_api_filter.h.

8.37.2.4 allow_in_place

```
int _GstTensorFilterFrameworkInfo::allow_in_place
```

TRUE(nonzero) if in-place transfer of input-to-output is allowed. Not supported in main, yet.

Definition at line 161 of file nnstreamer_plugin_api_filter.h.

8.37.2.5 hw_list

```
const accl_hw* _GstTensorFilterFrameworkInfo::hw_list
```

List of supported hardware accelerators by the framework. Positive response of this check does not guarantee successful running of model with this accelerator. Subplugin is supposed to allocate/deallocate.

Definition at line 165 of file nnstreamer_plugin_api_filter.h.

8.37.2.6 name

```
const char* _GstTensorFilterFrameworkInfo::name
```

Name of the neural network framework, searchable by FRAMEWORK property. Subplugin is supposed to allocate/deallocate.

Definition at line 160 of file nnstreamer_plugin_api_filter.h.

8.37.2.7 num_hw

```
int _GstTensorFilterFrameworkInfo::num_hw
```

number of hardware accelerators in the hw_list supported by the framework.

Definition at line 166 of file nnstreamer_plugin_api_filter.h.

8.37.2.8 run_without_model

```
int _GstTensorFilterFrameworkInfo::run_without_model
```

TRUE(nonzero) when the neural network framework does not need a model file. Tensor-filter will run invoke_NN without model.

Definition at line 163 of file nnstreamer_plugin_api_filter.h.

8.37.2.9 statistics

```
const GstTensorFilterFrameworkStatistics* _GstTensorFilterFrameworkInfo::statistics
```

usage statistics by the framework. This is shared across all opened instances of this framework.

Definition at line 169 of file nnstreamer_plugin_api_filter.h.

8.37.2.10 verify_model_path

```
int _GstTensorFilterFrameworkInfo::verify_model_path
```

TRUE(nonzero) when the NNS framework, not the sub-plugin, should verify the path of model files.

Definition at line 164 of file nnstreamer_plugin_api_filter.h.

The documentation for this struct was generated from the following file:

- nnstreamer/include/nnstreamer_plugin_api_filter.h

8.38 _GstTensorFilterFrameworkStatistics Struct Reference

Structure definition for tensor-filter framework statistics.

`#include <nnstreamer_plugin_api_filter.h>`

Public Attributes

- int64_t total_invoke_num
- int64_t total_invoke_latency
- int64_t total_overhead_latency

8.38.1 Detailed Description

Structure definition for tensor-filter framework statistics.

Definition at line 145 of file nnstreamer_plugin_api_filter.h.

8.38.2 Member Data Documentation

8.38.2.1 total_invoke_latency

`int64_t _GstTensorFilterFrameworkStatistics::total_invoke_latency`

The total accumulated invoke latency (usec)

Definition at line 148 of file nnstreamer_plugin_api_filter.h.

8.38.2.2 total_invoke_num

`int64_t _GstTensorFilterFrameworkStatistics::total_invoke_num`

The total number of calls to invoke

Definition at line 147 of file nnstreamer_plugin_api_filter.h.

8.38.2.3 total_overhead_latency

```
int64_t _GstTensorFilterFrameworkStatistics::total_overhead_latency
```

The total accumulated overhead latency from the extension (usec)

Definition at line 149 of file nnstreamer_plugin_api_filter.h.

The documentation for this struct was generated from the following file:

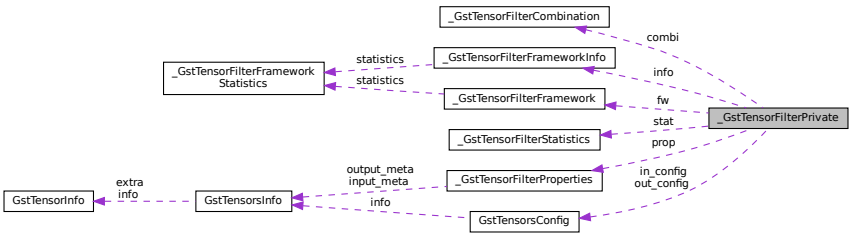
- nnstreamer/include/nnstreamer_plugin_api_filter.h

8.39 _GstTensorFilterPrivate Struct Reference

Structure definition for common tensor-filter properties.

```
#include <tensor_filter_common.h>
```

Collaboration diagram for _GstTensorFilterPrivate:



Public Attributes

- void * privateData
- GstTensorFilterProperties prop
- GstTensorFilterFrameworkInfo info
- GstTensorFilterStatistics stat
- const GstTensorFilterFramework * fw
- gboolean silent
- gboolean configured
- gboolean is_updatable
- gchar * config_path
- GstTensorsConfig in_config
- GstTensorsConfig out_config
- gint latency_mode
- gint throughput_mode
- gboolean latency_reporting
- gint64 latency_reported
- GstTensorFilterCombination combi
- nns_watchdog_h watchdog_h

8.39.1 Detailed Description

Structure definition for common tensor-filter properties.

Definition at line 152 of file tensor_filter_common.h.

8.39.2 Member Data Documentation

8.39.2.1 combi

```
GstTensorFilterCombination _GstTensorFilterPrivate::combi
```

Definition at line 173 of file tensor_filter_common.h.

8.39.2.2 config_path

```
gchar* _GstTensorFilterPrivate::config_path
```

Path to configuration file

Definition at line 164 of file tensor_filter_common.h.

8.39.2.3 configured

```
gboolean _GstTensorFilterPrivate::configured
```

True if already successfully configured tensor metadata

Definition at line 162 of file tensor_filter_common.h.

8.39.2.4 fw

```
const GstTensorFilterFramework* _GstTensorFilterPrivate::fw
```

The implementation core of the NNFW. NULL if not configured

Definition at line 158 of file tensor_filter_common.h.

8.39.2.5 in_config

```
GstTensorsConfig _GstTensorFilterPrivate::in_config
```

input tensor info

Definition at line 165 of file tensor_filter_common.h.

8.39.2.6 info

```
GstTensorFilterFrameworkInfo _GstTensorFilterPrivate::info
```

NNFW framework info

Definition at line 156 of file tensor_filter_common.h.

8.39.2.7 is_updatable

```
gboolean _GstTensorFilterPrivate::is_updatable
```

a given model to the filter is updatable if TRUE

Definition at line 163 of file tensor_filter_common.h.

8.39.2.8 latency_mode

```
gint _GstTensorFilterPrivate::latency_mode
```

latency profiling mode (0: off, 1: on, ...)

Definition at line 168 of file tensor_filter_common.h.

8.39.2.9 latency_reported

```
gint64 _GstTensorFilterPrivate::latency_reported
```

latency value reported (ns) in last LATENCY query

Definition at line 171 of file tensor_filter_common.h.

8.39.2.10 latency_reporting

```
gboolean _GstTensorFilterPrivate::latency_reporting
```

reporting of estimated filter latency is enabled

Definition at line 170 of file tensor_filter_common.h.

8.39.2.11 out_config

```
GstTensorsConfig _GstTensorFilterPrivate::out_config
```

output tensor info

Definition at line 166 of file tensor_filter_common.h.

8.39.2.12 privateData

```
void* _GstTensorFilterPrivate::privateData
```

NNFW plugin's private data is stored here

Definition at line 154 of file tensor_filter_common.h.

8.39.2.13 prop

```
GstTensorFilterProperties _GstTensorFilterPrivate::prop
```

NNFW plugin's properties

Definition at line 155 of file tensor_filter_common.h.

8.39.2.14 silent

```
gboolean _GstTensorFilterPrivate::silent
```

Verbose mode if FALSE. int instead of gboolean for non-glib custom plugins

Definition at line 161 of file tensor_filter_common.h.

8.39.2.15 stat

`GstTensorFilterStatistics _GstTensorFilterPrivate::stat`

NNFW plugin's statistics

Definition at line 157 of file tensor_filter_common.h.

8.39.2.16 throughput_mode

`gint _GstTensorFilterPrivate::throughput_mode`

throughput profiling mode (0: off, 1: on, ...)

Definition at line 169 of file tensor_filter_common.h.

8.39.2.17 watchdog_h

`nns_watchdog_h _GstTensorFilterPrivate::watchdog_h`

Watchdog to monitor occasional inputs

Definition at line 174 of file tensor_filter_common.h.

The documentation for this struct was generated from the following file:

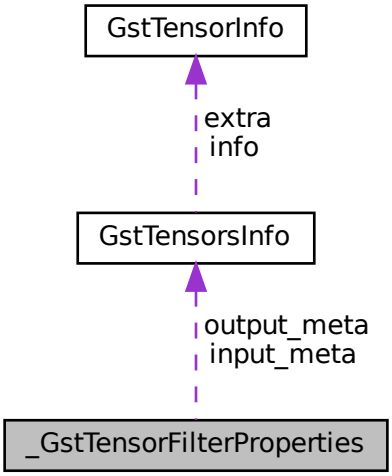
- nnstreamer/tensor_filter/tensor_filter_common.h

8.40 _GstTensorFilterProperties Struct Reference

GstTensorFilter's properties for NN framework (internal data structure)

`#include <nnstreamer_plugin_api_filter.h>`

Collaboration diagram for _GstTensorFilterProperties:



Public Attributes

- const char * fwname
- int fw_opened
- const char ** model_files
- int num_models
- int input_configured
- GstTensorsInfo input_meta
- tensors_layout input_layout
- unsigned int input_ranks [NNS_TENSOR_SIZE_LIMIT]
- int output_configured
- GstTensorsInfo output_meta
- tensors_layout output_layout
- unsigned int output_ranks [NNS_TENSOR_SIZE_LIMIT]
- const char * custom_properties
- accl_hw * hw_list
- int num_hw
- const char * accl_str
- char * shared_tensor_filter_key
- int latency
- int throughput
- int invoke_dynamic
- uint32_t suspend

8.40.1 Detailed Description

GstTensorFilter's properties for NN framework (internal data structure)

Because custom filters of tensor_filter may need to access internal data of GstTensorFilter, we define this data structure here.

Definition at line 112 of file nnstreamer_plugin_api_filter.h.

8.40.2 Member Data Documentation

8.40.2.1 accl_str

```
const char* _GstTensorFilterProperties::accl_str
```

accelerator configuration passed in as parameter, use in GstTensorFilterFramework V0 only

Definition at line 132 of file nnstreamer_plugin_api_filter.h.

8.40.2.2 custom_properties

```
const char* _GstTensorFilterProperties::custom_properties
```

sub-plugin specific custom property values in string

Definition at line 129 of file nnstreamer_plugin_api_filter.h.

8.40.2.3 fw_opened

```
int _GstTensorFilterProperties::fw_opened
```

TRUE IF open() is called or tried. Use int instead of gboolean because this is referred by custom plugins.

Definition at line 115 of file nnstreamer_plugin_api_filter.h.

8.40.2.4 fwname

```
const char* _GstTensorFilterProperties::fwname
```

The name of NN Framework

Definition at line 114 of file nnstreamer_plugin_api_filter.h.

8.40.2.5 hw_list

```
accl_hw* _GstTensorFilterProperties::hw_list
```

accelerators supported by framework intersected with user provided accelerator preference, use in GstTensor↩ FilterFramework V1 only

Definition at line 130 of file nnstreamer_plugin_api_filter.h.

8.40.2.6 input_configured

```
int _GstTensorFilterProperties::input_configured
```

TRUE if input tensor is configured. Use int instead of gboolean because this is referred by custom plugins.

Definition at line 119 of file nnstreamer_plugin_api_filter.h.

8.40.2.7 input_layout

```
tensors_layout _GstTensorFilterProperties::input_layout
```

data layout info provided as a property to tensor_filter for the input, defaults to _NNS_LAYOUT_ANY for all the tensors

Definition at line 121 of file nnstreamer_plugin_api_filter.h.

8.40.2.8 input_meta

```
GstTensorsInfo _GstTensorFilterProperties::input_meta
```

configured input tensor info

Definition at line 120 of file nnstreamer_plugin_api_filter.h.

8.40.2.9 input_ranks

```
unsigned int _GstTensorFilterProperties::input_ranks[NNS_TENSOR_SIZE_LIMIT]
```

the rank list of input tensors, it is calculated based on the dimension string.

Definition at line 122 of file nnstreamer_plugin_api_filter.h.

8.40.2.10 invoke_dynamic

```
int _GstTensorFilterProperties::invoke_dynamic
```

True for supporting invoke with flexible output.

Definition at line 137 of file nnstreamer_plugin_api_filter.h.

8.40.2.11 latency

```
int _GstTensorFilterProperties::latency
```

The average latency over the recent 10 inferences in microseconds

Definition at line 135 of file nnstreamer_plugin_api_filter.h.

8.40.2.12 model_files

```
const char** _GstTensorFilterProperties::model_files
```

File path to the model file (as an argument for NNFW). char instead of gchar for non-glib custom plugins

Definition at line 116 of file nnstreamer_plugin_api_filter.h.

8.40.2.13 num_hw

```
int _GstTensorFilterProperties::num_hw
```

number of hardware accelerators in the hw_list supported by the framework

Definition at line 131 of file nnstreamer_plugin_api_filter.h.

8.40.2.14 num_models

```
int _GstTensorFilterProperties::num_models
```

number of model files. Some frameworks need multiple model files to initialize the graph (caffe, caffe2)

Definition at line 117 of file nnstreamer_plugin_api_filter.h.

8.40.2.15 output_configured

```
int _GstTensorFilterProperties::output_configured
```

TRUE if output tensor is configured. Use int instead of gboolean because this is referred by custom plugins.

Definition at line 124 of file nnstreamer_plugin_api_filter.h.

8.40.2.16 output_layout

```
tensors_layout _GstTensorFilterProperties::output_layout
```

data layout info provided as a property to tensor_filter for the output, defaults to _NNS_LAYOUT_ANY for all the tensors

Definition at line 126 of file nnstreamer_plugin_api_filter.h.

8.40.2.17 output_meta

```
GstTensorsInfo _GstTensorFilterProperties::output_meta
```

configured output tensor info

Definition at line 125 of file nnstreamer_plugin_api_filter.h.

8.40.2.18 output_ranks

```
unsigned int _GstTensorFilterProperties::output_ranks[NNS_TENSOR_SIZE_LIMIT]
```

the rank list of output tensors, it is calculated based on the dimension string.

Definition at line 127 of file nnstreamer_plugin_api_filter.h.

8.40.2.19 shared_tensor_filter_key

```
char* _GstTensorFilterProperties::shared_tensor_filter_key
```

the shared instance key to use same model representation

Definition at line 133 of file nnstreamer_plugin_api_filter.h.

8.40.2.20 suspend

```
uint32_t _GstTensorFilterProperties::suspend
```

Timeout (ms) for suspend. (Unload the framework)

Definition at line 139 of file nnstreamer_plugin_api_filter.h.

8.40.2.21 throughput

```
int _GstTensorFilterProperties::throughput
```

The average throughput in the number of outputs per second

Definition at line 136 of file nnstreamer_plugin_api_filter.h.

The documentation for this struct was generated from the following file:

- nnstreamer/include/nnstreamer_plugin_api_filter.h

8.41 _GstTensorFilterStatistics Struct Reference

Structure definition for tensor-filter statistics.

`#include <tensor_filter_common.h>`

Public Attributes

- gint64 total_invoke_num
- gint64 total_invoke_latency
- gint64 old_total_invoke_num
- gint64 old_total_invoke_latency
- gint64 latest_invoke_time
- void * recent_latencies
- guint latency_ignore_count

8.41.1 Detailed Description

Structure definition for tensor-filter statistics.

Definition at line 117 of file tensor_filter_common.h.

8.41.2 Member Data Documentation

8.41.2.1 latency_ignore_count

`guint _GstTensorFilterStatistics::latency_ignore_count`

Definition at line 125 of file tensor_filter_common.h.

8.41.2.2 latest_invoke_time

`gint64 _GstTensorFilterStatistics::latest_invoke_time`

the latest invoke time (usec)

Definition at line 123 of file tensor_filter_common.h.

8.41.2.3 old_total_invoke_latency

```
gint64 _GstTensorFilterStatistics::old_total_invoke_latency
```

cached value. accumulated invoke latency (usec)

Definition at line 122 of file tensor_filter_common.h.

8.41.2.4 old_total_invoke_num

```
gint64 _GstTensorFilterStatistics::old_total_invoke_num
```

cached value. number of total invokes

Definition at line 121 of file tensor_filter_common.h.

8.41.2.5 recent_latencies

```
void* _GstTensorFilterStatistics::recent_latencies
```

data structure (e.g., queue) to hold recent latencies

Definition at line 124 of file tensor_filter_common.h.

8.41.2.6 total_invoke_latency

```
gint64 _GstTensorFilterStatistics::total_invoke_latency
```

accumulated invoke latency (usec)

Definition at line 120 of file tensor_filter_common.h.

8.41.2.7 total_invoke_num

```
gint64 _GstTensorFilterStatistics::total_invoke_num
```

number of total invokes

Definition at line 119 of file tensor_filter_common.h.

The documentation for this struct was generated from the following file:

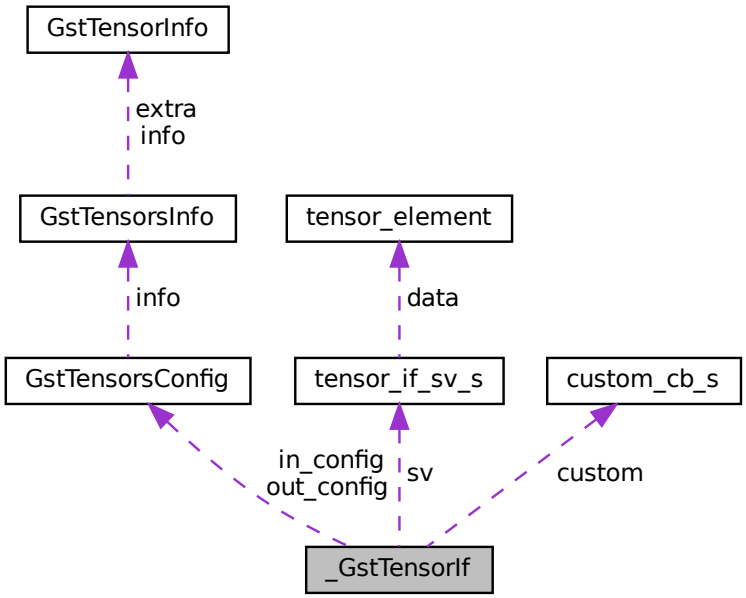
- nnstreamer/tensor_filter/tensor_filter_common.h

8.42 _GstTensorIf Struct Reference

Tensor If data structure.

`#include <gsttensor_if.h>`

Collaboration diagram for _GstTensorIf:



Public Attributes

- GstElement element
- GstPad * sinkpad
- GSList * srcpads
- gboolean silent
- GstTensorsConfig in_config
- GstTensorsConfig out_config [2]
- guint32 num_srcpads
- tensor_if_compared_value cv
- tensor_if_operator op
- tensor_if_behavior act_then
- tensor_if_behavior act_else
- tensor_if_sv_s sv [2]
- GList * cv_option
- GList * then_option
- GList * else_option
- gboolean custom_configured
- custom_cb_s custom
- GMutex lock

8.42.1 Detailed Description

Tensor If data structure.

Definition at line 122 of file gsttensor_if.h.

8.42.2 Member Data Documentation

8.42.2.1 act_else

`tensor_if_behavior _GstTensorIf::act_else`

Definition at line 136 of file gsttensor_if.h.

8.42.2.2 act_then

`tensor_if_behavior _GstTensorIf::act_then`

Definition at line 135 of file gsttensor_if.h.

8.42.2.3 custom

`custom_cb_s _GstTensorIf::custom`

Definition at line 143 of file gsttensor_if.h.

8.42.2.4 custom_configured

`gboolean _GstTensorIf::custom_configured`

Definition at line 142 of file gsttensor_if.h.

8.42.2.5 cv

`tensor_if_compared_value _GstTensorIf::cv`

compared value

Definition at line 133 of file gsttensor_if.h.

8.42.2.6 cv_option

```
GList* _GstTensorIf::cv_option
```

Definition at line 138 of file gsttensor_if.h.

8.42.2.7 element

```
GstElement _GstTensorIf::element
```

This is the parent object

Definition at line 124 of file gsttensor_if.h.

8.42.2.8 else_option

```
GList* _GstTensorIf::else_option
```

Definition at line 140 of file gsttensor_if.h.

8.42.2.9 in_config

```
GstTensorsConfig _GstTensorIf::in_config
```

input tensor info

Definition at line 129 of file gsttensor_if.h.

8.42.2.10 lock

```
GMutex _GstTensorIf::lock
```

Lock for custom callback

Definition at line 145 of file gsttensor_if.h.

8.42.2.11 num_srcpads

```
guint32 _GstTensorIf::num_srcpads
```

Definition at line 131 of file gsttensor_if.h.

8.42.2.12 op

```
tensor_if_operator _GstTensorIf::op
```

Definition at line 134 of file gsttensor_if.h.

8.42.2.13 out_config

```
GstTensorsConfig _GstTensorIf::out_config[2]
```

output tensor info

Definition at line 130 of file gsttensor_if.h.

8.42.2.14 silent

```
gboolean _GstTensorIf::silent
```

Definition at line 127 of file gsttensor_if.h.

8.42.2.15 sinkpad

```
GstPad* _GstTensorIf::sinkpad
```

Definition at line 125 of file gsttensor_if.h.

8.42.2.16 srcpads

```
GSList* _GstTensorIf::srcpads
```

Definition at line 126 of file gsttensor_if.h.

8.42.2.17 sv

`tensor_if_sv_s _GstTensorIf::sv[2]`

Definition at line 137 of file gsttensor_if.h.

8.42.2.18 then_option

`GList* _GstTensorIf::then_option`

Definition at line 139 of file gsttensor_if.h.

The documentation for this struct was generated from the following file:

- nnstreamer/elements/gsttensor_if.h

8.43 _GstTensorIfClass Struct Reference

GstTensorIfClass inherits GstElementClass.

`#include <gsttensor_if.h>`

Public Attributes

- GstElementClass parent_class

8.43.1 Detailed Description

GstTensorIfClass inherits GstElementClass.

Definition at line 151 of file gsttensor_if.h.

8.43.2 Member Data Documentation

8.43.2.1 parent_class

`GstElementClass _GstTensorIfClass::parent_class`

Inherits GstElementClass

Definition at line 153 of file gsttensor_if.h.

The documentation for this struct was generated from the following file:

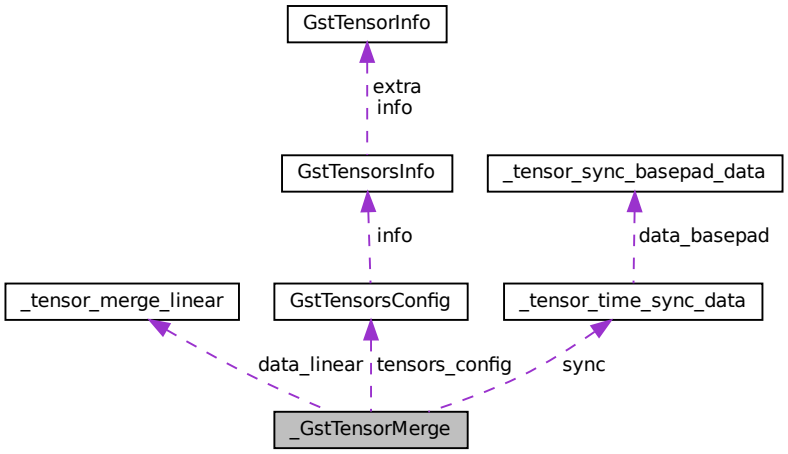
- nnstreamer/elements/gsttensor_if.h

8.44 _GstTensorMerge Struct Reference

Tensor Merge data structure.

`#include <gsttensor_merge.h>`

Collaboration diagram for _GstTensorMerge:



Public Attributes

- GstElement element
- gboolean silent
- tensor_time_sync_data sync
- GstPad * srcpad
- gchar * option
- tensor_merge_mode mode
- union {
  tensor_merge_linear data_linear
  };

- gboolean loaded
- GstCollectPads * collect
- gboolean negotiated
- gboolean need_segment
- gboolean need_stream_start
- gboolean send_stream_start
- GstClockTime current_time
- gboolean need_set_time
- GstTensorsConfig tensors_config

8.44.1 Detailed Description

Tensor Merge data structure.

Definition at line 71 of file gsttensor_merge.h.

8.44.2 Member Data Documentation

8.44.2.1 "@22

```
union { ... }
```

8.44.2.2 collect

```
GstCollectPads* _GstTensorMerge::collect
```

Definition at line 85 of file gsttensor_merge.h.

8.44.2.3 current_time

```
GstClockTime _GstTensorMerge::current_time
```

Definition at line 91 of file gsttensor_merge.h.

8.44.2.4 data_linear

```
tensor_merge_linear _GstTensorMerge::data_linear
```

Definition at line 81 of file gsttensor_merge.h.

8.44.2.5 element

```
GstElement _GstTensorMerge::element
```

Definition at line 73 of file gsttensor_merge.h.

8.44.2.6 loaded

```
gboolean _GstTensorMerge::loaded
```

Definition at line 84 of file gsttensor_merge.h.

8.44.2.7 mode

```
tensor_merge_mode _GstTensorMerge::mode
```

Definition at line 79 of file gsttensor_merge.h.

8.44.2.8 need_segment

```
gboolean _GstTensorMerge::need_segment
```

Definition at line 87 of file gsttensor_merge.h.

8.44.2.9 need_set_time

```
gboolean _GstTensorMerge::need_set_time
```

Definition at line 92 of file gsttensor_merge.h.

8.44.2.10 need_stream_start

```
gboolean _GstTensorMerge::need_stream_start
```

Definition at line 88 of file gsttensor_merge.h.

8.44.2.11 negotiated

```
gboolean _GstTensorMerge::negotiated
```

Definition at line 86 of file gsttensor_merge.h.

8.44.2.12 option

```
gchar* _GstTensorMerge::option
```

Definition at line 78 of file gsttensor_merge.h.

8.44.2.13 send_stream_start

```
gboolean _GstTensorMerge::send_stream_start
```

Definition at line 89 of file gsttensor_merge.h.

8.44.2.14 silent

```
gboolean _GstTensorMerge::silent
```

Definition at line 75 of file gsttensor_merge.h.

8.44.2.15 srcpad

```
GstPad* _GstTensorMerge::srcpad
```

Definition at line 77 of file gsttensor_merge.h.

8.44.2.16 sync

```
tensor_time_sync_data _GstTensorMerge::sync
```

Definition at line 76 of file gsttensor_merge.h.

8.44.2.17 tensors_config

```
GstTensorsConfig _GstTensorMerge::tensors_config
```

output tensors info

Definition at line 93 of file gsttensor_merge.h.

The documentation for this struct was generated from the following file:

- nnstreamer/elements/gsttensor_merge.h

8.45 _GstTensorMergeClass Struct Reference

GstTensorMergeClass inherits GstElementClass.

```
#include <gsttensor_merge.h>
```

Public Attributes

- GstElementClass parent_class

8.45.1 Detailed Description

GstTensorMergeClass inherits GstElementClass.

Definition at line 99 of file gsttensor_merge.h.

8.45.2 Member Data Documentation

8.45.2.1 parent_class

```
GstElementClass _GstTensorMergeClass::parent_class
```

Definition at line 101 of file gsttensor_merge.h.

The documentation for this struct was generated from the following file:

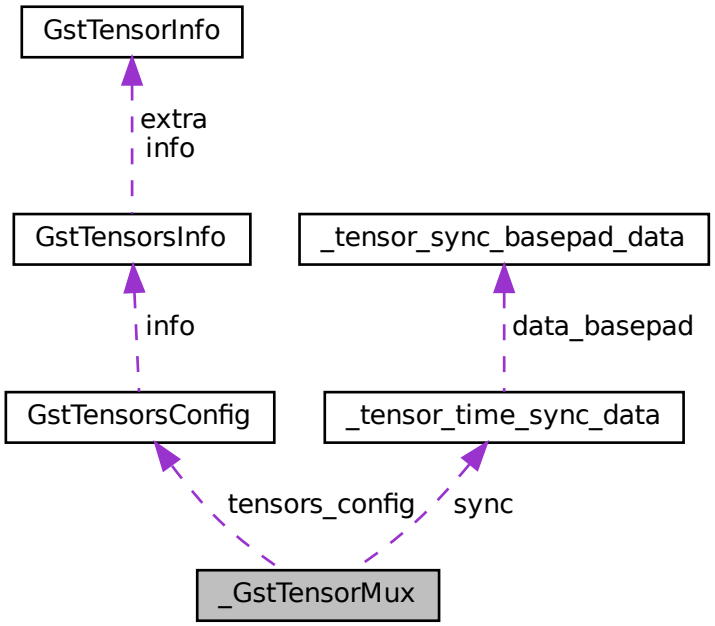
- nnstreamer/elements/gsttensor_merge.h

8.46 _GstTensorMux Struct Reference

Tensor Muxer data structure.

```
#include <gsttensor_mux.h>
```

Collaboration diagram for _GstTensorMux:



Public Attributes

- GstElement element
- gboolean silent
- tensor_time_sync_data sync
- GstPad * srcpad
- GstCollectPads * collect
- gboolean negotiated
- gboolean need_segment
- gboolean need_stream_start
- gboolean send_stream_start
- gboolean need_set_time
- GstClockTime current_time
- GstTensorsConfig tensors_config

8.46.1 Detailed Description

Tensor Muxer data structure.

Definition at line 48 of file gsttensor_mux.h.

8.46.2 Member Data Documentation

8.46.2.1 collect

```
GstCollectPads* _GstTensorMux::collect
```

Definition at line 56 of file gsttensor_mux.h.

8.46.2.2 current_time

```
GstClockTime _GstTensorMux::current_time
```

Definition at line 62 of file gsttensor_mux.h.

8.46.2.3 element

```
GstElement _GstTensorMux::element
```

Definition at line 50 of file gsttensor_mux.h.

8.46.2.4 need_segment

```
gboolean _GstTensorMux::need_segment
```

Definition at line 58 of file gsttensor_mux.h.

8.46.2.5 need_set_time

```
gboolean _GstTensorMux::need_set_time
```

Definition at line 61 of file gsttensor_mux.h.

8.46.2.6 need_stream_start

```
gboolean _GstTensorMux::need_stream_start
```

Definition at line 59 of file gsttensor_mux.h.

8.46.2.7 negotiated

```
gboolean _GstTensorMux::negotiated
```

Definition at line 57 of file gsttensor_mux.h.

8.46.2.8 send_stream_start

```
gboolean _GstTensorMux::send_stream_start
```

Definition at line 60 of file gsttensor_mux.h.

8.46.2.9 silent

```
gboolean _GstTensorMux::silent
```

Definition at line 52 of file gsttensor_mux.h.

8.46.2.10 srcpad

```
GstPad* _GstTensorMux::srcpad
```

Definition at line 54 of file gsttensor_mux.h.

8.46.2.11 sync

```
tensor_time_sync_data _GstTensorMux::sync
```

Definition at line 53 of file gsttensor_mux.h.

8.46.2.12 tensors_config

```
GstTensorsConfig _GstTensorMux::tensors_config
```

output tensors info

Definition at line 64 of file gsttensor_mux.h.

The documentation for this struct was generated from the following file:

- nnstreamer/elements/gsttensor_mux.h

8.47 _GstTensorMuxClass Struct Reference

GstTensroMuxClass inherits GstElementClass.

```
#include <gsttensor_mux.h>
```

Public Attributes

- GstElementClass parent_class

8.47.1 Detailed Description

GstTensroMuxClass inherits GstElementClass.

Definition at line 70 of file gsttensor_mux.h.

8.47.2 Member Data Documentation

8.47.2.1 parent_class

```
GstElementClass _GstTensorMuxClass::parent_class
```

Definition at line 72 of file gsttensor_mux.h.

The documentation for this struct was generated from the following file:

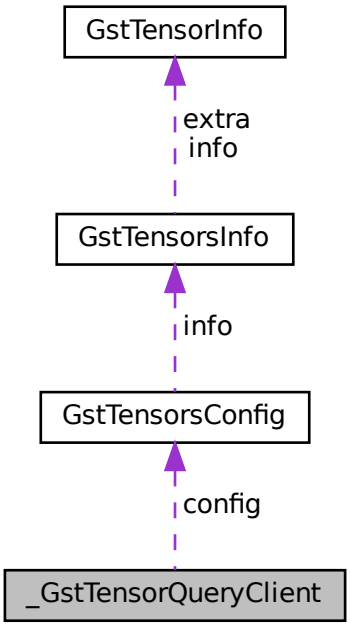
- nnstreamer/elements/gsttensor_mux.h

8.48 _GstTensorQueryClient Struct Reference

GstTensorQueryClient data structure.

```
#include <tensor_query_client.h>
```

Collaboration diagram for _GstTensorQueryClient:



Public Attributes

- GstElement element
- GstPad * sinkpad
- GstPad * srcpad
- gboolean silent
- gchar * in_caps_str
- gboolean is_tensor
- GstTensorsConfig config
- guint timeout
- gchar * topic
- gchar * host
- guint16 port
- gchar * dest_host
- guint16 dest_port
- nns_edge_connect_type_e connect_type
- nns_edge_h edge_h
- GAsyncQueue * msg_queue
- guint max_request
- guint requested_num

8.48.1 Detailed Description

GstTensorQueryClient data structure.

Definition at line 41 of file tensor_query_client.h.

8.48.2 Member Data Documentation

8.48.2.1 config

```
GstTensorsConfig _GstTensorQueryClient::config
```

Definition at line 50 of file tensor_query_client.h.

8.48.2.2 connect_type

```
nns_edge_connect_type_e _GstTensorQueryClient::connect_type
```

Definition at line 60 of file tensor_query_client.h.

8.48.2.3 dest_host

```
gchar* _GstTensorQueryClient::dest_host
```

Definition at line 57 of file tensor_query_client.h.

8.48.2.4 dest_port

```
guint16 _GstTensorQueryClient::dest_port
```

Definition at line 58 of file tensor_query_client.h.

8.48.2.5 edge_h

```
nns_edge_h _GstTensorQueryClient::edge_h
```

Definition at line 61 of file tensor_query_client.h.

8.48.2.6 element

```
GstElement _GstTensorQueryClient::element
```

parent object

Definition at line 43 of file tensor_query_client.h.

8.48.2.7 host

```
gchar* _GstTensorQueryClient::host
```

Definition at line 55 of file tensor_query_client.h.

8.48.2.8 in_caps_str

```
gchar* _GstTensorQueryClient::in_caps_str
```

Definition at line 48 of file tensor_query_client.h.

8.48.2.9 is_tensor

```
gboolean _GstTensorQueryClient::is_tensor
```

Definition at line 49 of file tensor_query_client.h.

8.48.2.10 max_request

```
guint _GstTensorQueryClient::max_request
```

Definition at line 64 of file tensor_query_client.h.

8.48.2.11 msg_queue

```
GAsyncQueue* _GstTensorQueryClient::msg_queue
```

Definition at line 62 of file tensor_query_client.h.

8.48.2.12 port

```
guint16 _GstTensorQueryClient::port
```

Definition at line 56 of file tensor_query_client.h.

8.48.2.13 requested_num

```
guint _GstTensorQueryClient::requested_num
```

Definition at line 65 of file tensor_query_client.h.

8.48.2.14 silent

```
gboolean _GstTensorQueryClient::silent
```

True if logging is minimized

Definition at line 47 of file tensor_query_client.h.

8.48.2.15 sinkpad

```
GstPad* _GstTensorQueryClient::sinkpad
```

sink pad

Definition at line 44 of file tensor_query_client.h.

8.48.2.16 srcpad

```
GstPad* _GstTensorQueryClient::srcpad
```

src pad

Definition at line 45 of file tensor_query_client.h.

8.48.2.17 timeout

```
guint _GstTensorQueryClient::timeout
```

timeout value (in ms) to wait message from server

Definition at line 51 of file tensor_query_client.h.

8.48.2.18 topic

```
gchar* _GstTensorQueryClient::topic
```

Main operation such as 'object_detection' or 'image_segmentation'

Definition at line 54 of file tensor_query_client.h.

The documentation for this struct was generated from the following file:

- nnstreamer/tensor_query/tensor_query_client.h

8.49 _GstTensorQueryClientClass Struct Reference

GstTensorQueryClientClass data structure.

```
#include <tensor_query_client.h>
```

Public Attributes

- GstElementClass parent_class

8.49.1 Detailed Description

GstTensorQueryClientClass data structure.

Definition at line 71 of file tensor_query_client.h.

8.49.2 Member Data Documentation

8.49.2.1 parent_class

```
GstElementClass _GstTensorQueryClientClass::parent_class
```

parent class

Definition at line 73 of file tensor_query_client.h.

The documentation for this struct was generated from the following file:

- nnstreamer/tensor_query/tensor_query_client.h

8.50 _GstTensorQueryServerSink Struct Reference

GstTensorQueryServerSink data structure.

```
#include <tensor_query_serversink.h>
```

Public Attributes

- GstBaseSink element
- guint sink_id
- guint timeout
- gint metaless_frame_limit
- gint metaless_frame_count
- nns_edge_connect_type_e connect_type

8.50.1 Detailed Description

GstTensorQueryServerSink data structure.

Definition at line 40 of file tensor_query_serversink.h.

8.50.2 Member Data Documentation

8.50.2.1 connect_type

```
nns_edge_connect_type_e _GstTensorQueryServerSink::connect_type
```

Definition at line 49 of file tensor_query_serversink.h.

8.50.2.2 element

```
GstBaseSink _GstTensorQueryServerSink::element
```

parent object

Definition at line 42 of file tensor_query_serversink.h.

8.50.2.3 metaless_frame_count

```
gint _GstTensorQueryServerSink::metaless_frame_count
```

Definition at line 47 of file tensor_query_serversink.h.

8.50.2.4 metaless_frame_limit

```
gint _GstTensorQueryServerSink::metaless_frame_limit
```

Definition at line 46 of file tensor_query_serversink.h.

8.50.2.5 sink_id

```
guint _GstTensorQueryServerSink::sink_id
```

Definition at line 43 of file tensor_query_serversink.h.

8.50.2.6 timeout

```
guint _GstTensorQueryServerSink::timeout
```

Definition at line 45 of file tensor_query_serversink.h.

The documentation for this struct was generated from the following file:

- nnstreamer/tensor_query/tensor_query_serversink.h

8.51 _GstTensorQueryServerSinkClass Struct Reference

GstTensorQueryServerSinkClass data structure.

```
#include <tensor_query_serversink.h>
```

Public Attributes

- GstBaseSinkClass parent_class

8.51.1 Detailed Description

GstTensorQueryServerSinkClass data structure.

Definition at line 55 of file tensor_query_serversink.h.

8.51.2 Member Data Documentation

8.51.2.1 parent_class

```
GstBaseSinkClass _GstTensorQueryServerSinkClass::parent_class
```

parent class

Definition at line 57 of file tensor_query_serversink.h.

The documentation for this struct was generated from the following file:

- nnstreamer/tensor_query/tensor_query_serversink.h

8.52 _GstTensorQueryServerSrc Struct Reference

GstTensorQueryServerSrc data structure.

`#include <tensor_query_serversrc.h>`

Public Attributes

- GstPushSrc element
- guint src_id
- gboolean configured
- gchar * host
- guint16 port
- gchar * dest_host
- guint16 dest_port
- guint timeout
- gchar * topic
- nns_edge_connect_type_e connect_type
- GAsyncQueue * msg_queue
- gboolean playing

8.52.1 Detailed Description

GstTensorQueryServerSrc data structure.

Definition at line 39 of file tensor_query_serversrc.h.

8.52.2 Member Data Documentation

8.52.2.1 configured

`gboolean _GstTensorQueryServerSrc::configured`

Definition at line 43 of file tensor_query_serversrc.h.

8.52.2.2 connect_type

`nns_edge_connect_type_e _GstTensorQueryServerSrc::connect_type`

Definition at line 54 of file tensor_query_serversrc.h.

8.52.2.3 dest_host

```
gchar* _GstTensorQueryServerSrc::dest_host
```

Definition at line 47 of file tensor_query_serversrc.h.

8.52.2.4 dest_port

```
guint16 _GstTensorQueryServerSrc::dest_port
```

Definition at line 48 of file tensor_query_serversrc.h.

8.52.2.5 element

```
GstPushSrc _GstTensorQueryServerSrc::element
```

Definition at line 41 of file tensor_query_serversrc.h.

8.52.2.6 host

```
gchar* _GstTensorQueryServerSrc::host
```

Definition at line 45 of file tensor_query_serversrc.h.

8.52.2.7 msg_queue

```
GAsyncQueue* _GstTensorQueryServerSrc::msg_queue
```

Definition at line 55 of file tensor_query_serversrc.h.

8.52.2.8 playing

```
gboolean _GstTensorQueryServerSrc::playing
```

Definition at line 56 of file tensor_query_serversrc.h.

8.52.2.9 port

```
guint16 _GstTensorQueryServerSrc::port
```

Definition at line 46 of file tensor_query_serversrc.h.

8.52.2.10 src_id

```
guint _GstTensorQueryServerSrc::src_id
```

Definition at line 42 of file tensor_query_serversrc.h.

8.52.2.11 timeout

```
guint _GstTensorQueryServerSrc::timeout
```

Definition at line 49 of file tensor_query_serversrc.h.

8.52.2.12 topic

```
gchar* _GstTensorQueryServerSrc::topic
```

Main operation such as 'object_detection' or 'image_segmentation'

Definition at line 52 of file tensor_query_serversrc.h.

The documentation for this struct was generated from the following file:

- nnstreamer/tensor_query/tensor_query_serversrc.h

8.53 _GstTensorQueryServerSrcClass Struct Reference

GstTensorQueryServerSrcClass data structure.

```
#include <tensor_query_serversrc.h>
```

Public Attributes

- GstPushSrcClass parent_class

8.53.1 Detailed Description

GstTensorQueryServerSrcClass data structure.

Definition at line 62 of file tensor_query_serversrc.h.

8.53.2 Member Data Documentation

8.53.2.1 parent_class

```
GstPushSrcClass _GstTensorQueryServerSrcClass::parent_class
```

parent class

Definition at line 64 of file tensor_query_serversrc.h.

The documentation for this struct was generated from the following file:

- nnstreamer/tensor_query/tensor_query_serversrc.h

8.54 _GstTensorRate Struct Reference

Tensor Rate data structure.

```
#include <gsttensor_rate.h>
```

Public Attributes

- GstBaseTransform element
- GstBuffer * prevbuf
- GstSegment segment
- guint64 out_frame_count
- gboolean sent_qos_on_passthrough
- gint from_rate_numerator
- gint from_rate_denominator
- gint to_rate_numerator
- gint to_rate_denominator
- guint64 base_ts
- guint64 prev_ts
- guint64 next_ts
- guint64 last_ts
- guint64 in
- guint64 out
- guint64 dup
- guint64 drop
- gint rate_n
- gint rate_d
- gboolean silent
- gboolean throttle

8.54.1 Detailed Description

Tensor Rate data structure.

Definition at line 37 of file gsttensor_rate.h.

8.54.2 Member Data Documentation

8.54.2.1 base_ts

```
guint64 _GstTensorRate::base_ts
```

Timestamp used in next_ts calculation

Definition at line 55 of file gsttensor_rate.h.

8.54.2.2 drop

```
guint64 _GstTensorRate::drop
```

stat property

Definition at line 61 of file gsttensor_rate.h.

8.54.2.3 dup

```
guint64 _GstTensorRate::dup
```

Definition at line 61 of file gsttensor_rate.h.

8.54.2.4 element

```
GstBaseTransform _GstTensorRate::element
```

This is the parent object

Definition at line 39 of file gsttensor_rate.h.

8.54.2.5 from_rate_denominator

```
gint _GstTensorRate::from_rate_denominator
```

framerate denominator (From)

Definition at line 49 of file gsttensor_rate.h.

8.54.2.6 from_rate_numerator

```
gint _GstTensorRate::from_rate_numerator
```

Caps negotiation framerate numerator (From)

Definition at line 48 of file gsttensor_rate.h.

8.54.2.7 in

```
guint64 _GstTensorRate::in
```

Properties

Definition at line 61 of file gsttensor_rate.h.

8.54.2.8 last_ts

```
guint64 _GstTensorRate::last_ts
```

Timestamp of last input buffer

Definition at line 58 of file gsttensor_rate.h.

8.54.2.9 next_ts

```
guint64 _GstTensorRate::next_ts
```

Timestamp of next buffer to output

Definition at line 57 of file gsttensor_rate.h.

8.54.2.10 out

```
guint64 _GstTensorRate::out
```

Definition at line 61 of file gsttensor_rate.h.

8.54.2.11 out_frame_count

```
guint64 _GstTensorRate::out_frame_count
```

number of frames output

Definition at line 43 of file gsttensor_rate.h.

8.54.2.12 prev_ts

```
guint64 _GstTensorRate::prev_ts
```

Previous buffer timestamp

Definition at line 56 of file gsttensor_rate.h.

8.54.2.13 prevbuf

```
GstBuffer* _GstTensorRate::prevbuf
```

previous buffer

Definition at line 41 of file gsttensor_rate.h.

8.54.2.14 rate_d

```
gint _GstTensorRate::rate_d
```

framerate property

Definition at line 62 of file gsttensor_rate.h.

8.54.2.15 rate_n

```
gint _GstTensorRate::rate_n
```

Definition at line 62 of file gsttensor_rate.h.

8.54.2.16 segment

```
GstSegment _GstTensorRate::segment
```

current segment

Definition at line 42 of file gsttensor_rate.h.

8.54.2.17 sent_qos_on_passthrough

```
gboolean _GstTensorRate::sent_qos_on_passthrough
```

qos event on passthrough

Definition at line 45 of file gsttensor_rate.h.

8.54.2.18 silent

```
gboolean _GstTensorRate::silent
```

debug property

Definition at line 63 of file gsttensor_rate.h.

8.54.2.19 throttle

```
gboolean _GstTensorRate::throttle
```

throttle property

Definition at line 64 of file gsttensor_rate.h.

8.54.2.20 to_rate_denominator

```
gint _GstTensorRate::to_rate_denominator
```

framerate denominator (To)

Definition at line 52 of file gsttensor_rate.h.

8.54.2.21 to_rate_numerator

```
gint _GstTensorRate::to_rate_numerator
```

framerate numerator (To)

Definition at line 51 of file gsttensor_rate.h.

The documentation for this struct was generated from the following file:

- nnstreamer/elements/gsttensor_rate.h

8.55 _GstTensorRateClass Struct Reference

GstTensorRateClass inherits GstElementClass.

```
#include <gsttensor_rate.h>
```

Public Attributes

- GstBaseTransformClass parent_class

8.55.1 Detailed Description

GstTensorRateClass inherits GstElementClass.

Definition at line 70 of file gsttensor_rate.h.

8.55.2 Member Data Documentation

8.55.2.1 parent_class

```
GstBaseTransformClass _GstTensorRateClass::parent_class
```

Inherits GstBaseTransformClass

Definition at line 72 of file gsttensor_rate.h.

The documentation for this struct was generated from the following file:

- nnstreamer/elements/gsttensor_rate.h

8.56 _GstTensorRepoSink Struct Reference

GstTensorRepoSink data structure.

```
#include <gsttensor_reposink.h>
```

Public Attributes

- GstBaseSink element
- gboolean silent
- guint signal_rate
- GstClockTime last_render_time
- GstCaps * in_caps
- gboolean set_startid
- guint myid
- guint o_myid

8.56.1 Detailed Description

GstTensorRepoSink data structure.

GstTensorRepoSink inherits GstBaseSink.

Definition at line 54 of file gsttensor_reposink.h.

8.56.2 Member Data Documentation

8.56.2.1 element

```
GstBaseSink _GstTensorRepoSink::element
```

Definition at line 56 of file gsttensor_reposink.h.

8.56.2.2 in_caps

```
GstCaps* _GstTensorRepoSink::in_caps
```

Definition at line 61 of file gsttensor_reposink.h.

8.56.2.3 last_render_time

```
GstClockTime _GstTensorRepoSink::last_render_time
```

Definition at line 60 of file gsttensor_reposink.h.

8.56.2.4 myid

```
guint _GstTensorRepoSink::myid
```

Definition at line 63 of file gsttensor_reposink.h.

8.56.2.5 o_myid

```
guint _GstTensorRepoSink::o_myid
```

Definition at line 64 of file gsttensor_reposink.h.

8.56.2.6 set_startid

```
gboolean _GstTensorRepoSink::set_startid
```

Definition at line 62 of file gsttensor_reposink.h.

8.56.2.7 signal_rate

```
guint _GstTensorRepoSink::signal_rate
```

Definition at line 59 of file gsttensor_reposink.h.

8.56.2.8 silent

```
gboolean _GstTensorRepoSink::silent
```

Definition at line 58 of file gsttensor_reposink.h.

The documentation for this struct was generated from the following file:

- nnstreamer/elements/gsttensor_reposink.h

8.57 _GstTensorRepoSinkClass Struct Reference

GstTensorRepoSinkClass data structure.

```
#include <gsttensor_reposink.h>
```

Public Attributes

- GstBaseSinkClass parent_class

8.57.1 Detailed Description

GstTensorRepoSinkClass data structure.

GstTensorRepoSink inherits GstBaseSink.

Definition at line 72 of file gsttensor_reposink.h.

8.57.2 Member Data Documentation

8.57.2.1 parent_class

```
GstBaseSinkClass _GstTensorRepoSinkClass::parent_class
```

Definition at line 74 of file gsttensor_reposink.h.

The documentation for this struct was generated from the following file:

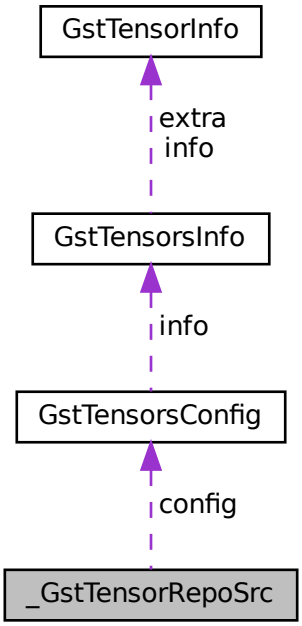
- nnstreamer/elements/gsttensor_reposink.h

8.58 _GstTensorRepoSrc Struct Reference

GstTensorRepoSrc data structure.

`#include <gsttensor_reposrc.h>`

Collaboration diagram for _GstTensorRepoSrc:



Public Attributes

- GstPushSrc parent
- GstTensorsConfig config
- gboolean silent
- guint myid
- guint o_myid
- GstCaps * caps
- gboolean ini
- gint fps_n
- gint fps_d
- gboolean negotiation
- gboolean set_startid

8.58.1 Detailed Description

GstTensorRepoSrc data structure.

GstTensorRepoSrc inherits GstPushSrc

Definition at line 54 of file gsttensor_reposrc.h.

8.58.2 Member Data Documentation

8.58.2.1 caps

```
GstCaps* _GstTensorRepoSrc::caps
```

Definition at line 61 of file gsttensor_reposrc.h.

8.58.2.2 config

```
GstTensorsConfig _GstTensorRepoSrc::config
```

Definition at line 57 of file gsttensor_reposrc.h.

8.58.2.3 fps_d

```
gint _GstTensorRepoSrc::fps_d
```

Definition at line 64 of file gsttensor_reposrc.h.

8.58.2.4 fps_n

```
gint _GstTensorRepoSrc::fps_n
```

Definition at line 63 of file gsttensor_reposrc.h.

8.58.2.5 ini

```
gboolean _GstTensorRepoSrc::ini
```

Definition at line 62 of file gsttensor_reposrc.h.

8.58.2.6 myid

```
guint _GstTensorRepoSrc::myid
```

Definition at line 59 of file gsttensor_reposrc.h.

8.58.2.7 negotiation

```
gboolean _GstTensorRepoSrc::negotiation
```

Definition at line 65 of file gsttensor_reposrc.h.

8.58.2.8 o_myid

```
guint _GstTensorRepoSrc::o_myid
```

Definition at line 60 of file gsttensor_reposrc.h.

8.58.2.9 parent

```
GstPushSrc _GstTensorRepoSrc::parent
```

Definition at line 56 of file gsttensor_reposrc.h.

8.58.2.10 set_startid

```
gboolean _GstTensorRepoSrc::set_startid
```

Definition at line 66 of file gsttensor_reposrc.h.

8.58.2.11 silent

```
gboolean _GstTensorRepoSrc::silent
```

Definition at line 58 of file gsttensor_reposrc.h.

The documentation for this struct was generated from the following file:

- nnstreamer/elements/gsttensor_reposrc.h

8.59 _GstTensorRepoSrcClass Struct Reference

GstTensorRepoSrcClass data structure.

`#include <gsttensor_reposrc.h>`

Public Attributes

- GstPushSrcClass parent_class

8.59.1 Detailed Description

GstTensorRepoSrcClass data structure.

GstTensorRepoSrc inherits GstPushSrc

Definition at line 74 of file gsttensor_reposrc.h.

8.59.2 Member Data Documentation

8.59.2.1 parent_class

`GstPushSrcClass _GstTensorRepoSrcClass::parent_class`

Definition at line 76 of file gsttensor_reposrc.h.

The documentation for this struct was generated from the following file:

- nnstreamer/elements/gsttensor_reposrc.h

8.60 _GstTensorSink Struct Reference

GstTensorSink data structure.

`#include <gsttensor_sink.h>`

Public Attributes

- GstBaseSink element
- GMutex mutex
- gboolean silent
- gboolean emit_signal
- guint signal_rate
- GstClockTime last_render_time

8.60.1 Detailed Description

GstTensorSink data structure.

GstTensorSink inherits GstBaseSink.

Definition at line 55 of file gsttensor_sink.h.

8.60.2 Member Data Documentation

8.60.2.1 element

`GstBaseSink _GstTensorSink::element`

parent object

Definition at line 57 of file gsttensor_sink.h.

8.60.2.2 emit_signal

`gboolean _GstTensorSink::emit_signal`

true to emit signal for new data, eos

Definition at line 61 of file gsttensor_sink.h.

8.60.2.3 last_render_time

`GstClockTime _GstTensorSink::last_render_time`

buffer rendered time

Definition at line 63 of file gsttensor_sink.h.

8.60.2.4 mutex

`GMutex _GstTensorSink::mutex`

mutex for processing

Definition at line 59 of file gsttensor_sink.h.

8.60.2.5 signal_rate

```
guint _GstTensorSink::signal_rate
```

new data signals per second

Definition at line 62 of file gsttensor_sink.h.

8.60.2.6 silent

```
gboolean _GstTensorSink::silent
```

true to print minimized log

Definition at line 60 of file gsttensor_sink.h.

The documentation for this struct was generated from the following file:

- nnstreamer/elements/gsttensor_sink.h

8.61 _GstTensorSinkClass Struct Reference

GstTensorSinkClass data structure.

```
#include <gsttensor_sink.h>
```

Public Attributes

- GstBaseSinkClass parent_class
- void(∗ new_data)(GstElement ∗element, GstBuffer ∗buffer)
- void(∗ stream_start)(GstElement ∗element)
- void(∗ eos)(GstElement ∗element)

8.61.1 Detailed Description

GstTensorSinkClass data structure.

GstTensorSink inherits GstBaseSink.

Definition at line 71 of file gsttensor_sink.h.

8.61.2 Member Data Documentation

8.61.2.1 eos

```
void(* _GstTensorSinkClass::eos) (GstElement *element)
```

signal when end of stream reached

Definition at line 78 of file gsttensor_sink.h.

8.61.2.2 new_data

```
void(* _GstTensorSinkClass::new_data) (GstElement *element, GstBuffer *buffer)
```

signals signal when new data received

Definition at line 76 of file gsttensor_sink.h.

8.61.2.3 parent_class

```
GstBaseSinkClass _GstTensorSinkClass::parent_class
```

parent class

Definition at line 73 of file gsttensor_sink.h.

8.61.2.4 stream_start

```
void(* _GstTensorSinkClass::stream_start) (GstElement *element)
```

signal when stream started

Definition at line 77 of file gsttensor_sink.h.

The documentation for this struct was generated from the following file:

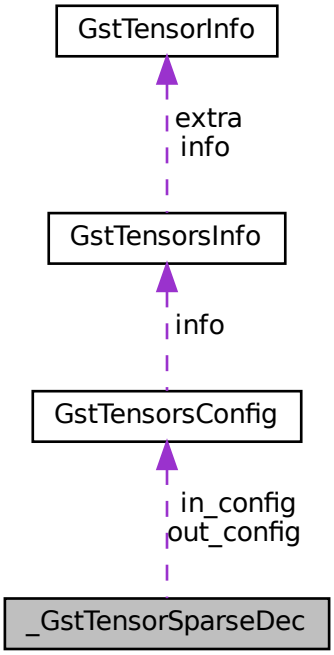
- nnstreamer/elements/gsttensor_sink.h

8.62 _GstTensorSparseDec Struct Reference

GstTensorSparseDec data structure.

`#include <gsttensor_sparsedec.h>`

Collaboration diagram for _GstTensorSparseDec:



Public Attributes

- GstElement element
- GstPad * sinkpad
- GstPad * srcpad
- GstTensorsConfig in_config
- GstTensorsConfig out_config
- gboolean silent

8.62.1 Detailed Description

GstTensorSparseDec data structure.

Definition at line 39 of file gsttensor_sparsedec.h.

8.62.2 Member Data Documentation

8.62.2.1 element

```
GstElement _GstTensorSparseDec::element
```

parent object

Definition at line 41 of file gsttensor_sparsedec.h.

8.62.2.2 in_config

```
GstTensorsConfig _GstTensorSparseDec::in_config
```

input tensors config

Definition at line 46 of file gsttensor_sparsedec.h.

8.62.2.3 out_config

```
GstTensorsConfig _GstTensorSparseDec::out_config
```

output tensors config

Definition at line 47 of file gsttensor_sparsedec.h.

8.62.2.4 silent

```
gboolean _GstTensorSparseDec::silent
```

true to print minimized log

Definition at line 48 of file gsttensor_sparsedec.h.

8.62.2.5 sinkpad

```
GstPad* _GstTensorSparseDec::sinkpad
```

sink pad

Definition at line 42 of file gsttensor_sparsedec.h.

8.62.2.6 srcpad

```
GstPad* _GstTensorSparseDec::srcpad
```

src pad

Definition at line 43 of file gsttensor_sparsedec.h.

The documentation for this struct was generated from the following file:

- nnstreamer/elements/gsttensor_sparsedec.h

8.63 _GstTensorSparseDecClass Struct Reference

GstTensorSparseClass data structure.

```
#include <gsttensor_sparsedec.h>
```

Public Attributes

- GstElementClass parent_class

8.63.1 Detailed Description

GstTensorSparseClass data structure.

Definition at line 54 of file gsttensor_sparsedec.h.

8.63.2 Member Data Documentation

8.63.2.1 parent_class

```
GstElementClass _GstTensorSparseDecClass::parent_class
```

parent class

Definition at line 56 of file gsttensor_sparsedec.h.

The documentation for this struct was generated from the following file:

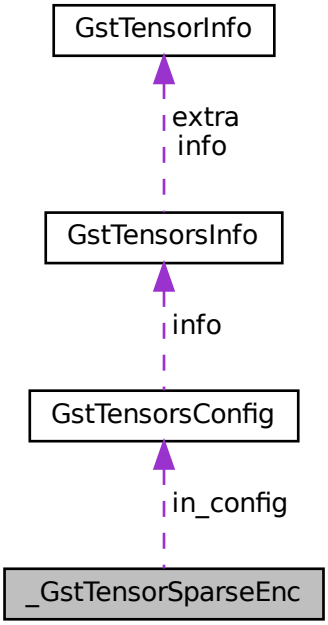
- nnstreamer/elements/gsttensor_sparsedec.h

8.64 _GstTensorSparseEnc Struct Reference

GstTensorSparseEnc data structure.

`#include <gsttensor_sparseenc.h>`

Collaboration diagram for _GstTensorSparseEnc:



Public Attributes

- GstElement element
- GstPad * sinkpad
- GstPad * srcpad
- GstTensorsConfig in_config
- gboolean silent

8.64.1 Detailed Description

GstTensorSparseEnc data structure.

Definition at line 39 of file gsttensor_sparseenc.h.

8.64.2 Member Data Documentation

8.64.2.1 element

```
GstElement _GstTensorSparseEnc::element
```

parent object

Definition at line 41 of file gsttensor_sparseenc.h.

8.64.2.2 in_config

```
GstTensorsConfig _GstTensorSparseEnc::in_config
```

input tensors config

Definition at line 46 of file gsttensor_sparseenc.h.

8.64.2.3 silent

```
gboolean _GstTensorSparseEnc::silent
```

true to print minimized log

Definition at line 47 of file gsttensor_sparseenc.h.

8.64.2.4 sinkpad

```
GstPad* _GstTensorSparseEnc::sinkpad
```

sink pad

Definition at line 42 of file gsttensor_sparseenc.h.

8.64.2.5 srcpad

```
GstPad* _GstTensorSparseEnc::srcpad
```

src pad

Definition at line 43 of file gsttensor_sparseenc.h.

The documentation for this struct was generated from the following file:

- nnstreamer/elements/gsttensor_sparseenc.h

8.65 _GstTensorSparseEncClass Struct Reference

GstTensorSparseClass data structure.

```
#include <gsttensor_sparseenc.h>
```

Public Attributes

- GstElementClass parent_class

8.65.1 Detailed Description

GstTensorSparseClass data structure.

Definition at line 53 of file gsttensor_sparseenc.h.

8.65.2 Member Data Documentation

8.65.2.1 parent_class

```
GstElementClass _GstTensorSparseEncClass::parent_class
```

parent class

Definition at line 55 of file gsttensor_sparseenc.h.

The documentation for this struct was generated from the following file:

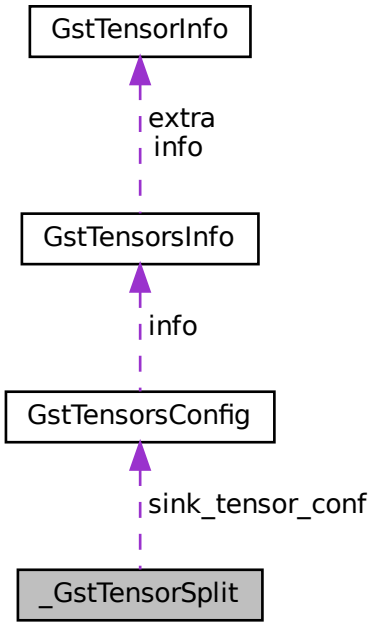
- nnstreamer/elements/gsttensor_sparseenc.h

8.66 _GstTensorSplit Struct Reference

Tensor Split data structure.

`#include <gsttensor_split.h>`

Collaboration diagram for _GstTensorSplit:



Public Attributes

- GstElement element
- gboolean silent
- GstPad * sinkpad
- GSList * srcpads
- guint32 num_tensors
- guint32 num_srcpads
- GList * tensorpick
- GArray * tensorseg
- gboolean have_group_id
- guint group_id
- GstTensorsConfig sink_tensor_conf

8.66.1 Detailed Description

Tensor Split data structure.

Definition at line 50 of file gsttensor_split.h.

8.66.2 Member Data Documentation

8.66.2.1 element

```
GstElement _GstTensorSplit::element
```

Definition at line 52 of file gsttensor_split.h.

8.66.2.2 group_id

```
guint _GstTensorSplit::group_id
```

Definition at line 62 of file gsttensor_split.h.

8.66.2.3 have_group_id

```
gboolean _GstTensorSplit::have_group_id
```

Definition at line 61 of file gsttensor_split.h.

8.66.2.4 num_srcpads

```
guint32 _GstTensorSplit::num_srcpads
```

Definition at line 58 of file gsttensor_split.h.

8.66.2.5 num_tensors

```
guint32 _GstTensorSplit::num_tensors
```

Definition at line 57 of file gsttensor_split.h.

8.66.2.6 silent

```
gboolean _GstTensorSplit::silent
```

Definition at line 54 of file gsttensor_split.h.

8.66.2.7 sink_tensor_conf

```
GstTensorsConfig _GstTensorSplit::sink_tensor_conf
```

Definition at line 63 of file gsttensor_split.h.

8.66.2.8 sinkpad

```
GstPad* _GstTensorSplit::sinkpad
```

Definition at line 55 of file gsttensor_split.h.

8.66.2.9 srcpads

```
GSList* _GstTensorSplit::srcpads
```

Definition at line 56 of file gsttensor_split.h.

8.66.2.10 tensorpick

```
GList* _GstTensorSplit::tensorpick
```

Definition at line 59 of file gsttensor_split.h.

8.66.2.11 tensorseg

```
GArray* _GstTensorSplit::tensorseg
```

Definition at line 60 of file gsttensor_split.h.

The documentation for this struct was generated from the following file:

- nnstreamer/elements/gsttensor_split.h

8.67 _GstTensorSplitClass Struct Reference

GstTensorSplitClass inherits GstElementClass.

`#include <gsttensor_split.h>`

Public Attributes

- GstElementClass parent_class

8.67.1 Detailed Description

GstTensorSplitClass inherits GstElementClass.

Definition at line 69 of file gsttensor_split.h.

8.67.2 Member Data Documentation

8.67.2.1 parent_class

`GstElementClass _GstTensorSplitClass::parent_class`

Definition at line 71 of file gsttensor_split.h.

The documentation for this struct was generated from the following file:

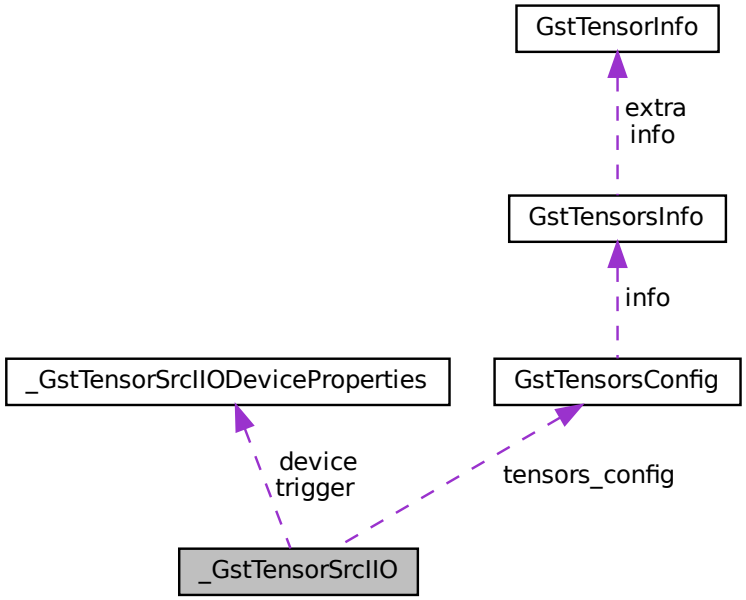
- nnstreamer/elements/gsttensor_split.h

8.68 _GstTensorSrcIIO Struct Reference

GstTensorSrcIIO data structure.

`#include <gsttensor_srciio.h>`

Collaboration diagram for _GstTensorSrcIIO:



Public Attributes

- GstBaseSrc element
- gboolean silent
- gboolean configured
- gchar * mode
- gchar * base_dir
- gchar * dev_dir
- GstTensorSrcIIODeviceProperties device
- GstTensorSrcIIODeviceProperties trigger
- GList * channels
- GHashTable * custom_channel_table
- channels_enabled_options channels_enabled
- guint scan_size
- struct pollfd * buffer_data_fp
- guint num_channels_enabled
- gboolean merge_channels_data
- gboolean is_tensor
- guint buffer_capacity
- guint64 sampling_frequency
- guint64 default_sampling_frequency
- guint default_buffer_capacity
- gchar * default_trigger
- gint poll_timeout
- GstTensorsConfig * tensors_config

8.68.1 Detailed Description

GstTensorSrcIIO data structure.

GstTensorSrcIIO inherits GstBaseSrcIIO.

Definition at line 104 of file gsttensor_srciio.h.

8.68.2 Member Data Documentation

8.68.2.1 base_dir

```
gchar* _GstTensorSrcIIO::base_dir
```

Base directory for IIO devices

Definition at line 114 of file gsttensor_srciio.h.

8.68.2.2 buffer_capacity

```
guint _GstTensorSrcIIO::buffer_capacity
```

size of the buffer

Definition at line 126 of file gsttensor_srciio.h.

8.68.2.3 buffer_data_fp

```
struct pollfd* _GstTensorSrcIIO::buffer_data_fp
```

pollfd for reading data buffer

Definition at line 122 of file gsttensor_srciio.h.

8.68.2.4 channels

```
GList* _GstTensorSrcIIO::channels
```

list of enabled channels

Definition at line 118 of file gsttensor_srciio.h.

8.68.2.5 channels_enabled

```
channels_enabled_options _GstTensorSrcIIO::channels_enabled
```

enabling which channels

Definition at line 120 of file gsttensor_srciio.h.

8.68.2.6 configured

```
gboolean _GstTensorSrcIIO::configured
```

true if device is configured and ready

Definition at line 110 of file gsttensor_srciio.h.

8.68.2.7 custom_channel_table

```
GHashTable* _GstTensorSrcIIO::custom_channel_table
```

table of idx of channels to be enabled

Definition at line 119 of file gsttensor_srciio.h.

8.68.2.8 default_buffer_capacity

```
guint _GstTensorSrcIIO::default_buffer_capacity
```

size of the buffer

Definition at line 130 of file gsttensor_srciio.h.

8.68.2.9 default_sampling_frequency

```
guint64 _GstTensorSrcIIO::default_sampling_frequency
```

default set value of sampling frequency

Definition at line 129 of file gsttensor_srciio.h.

8.68.2.10 default_trigger

```
gchar* _GstTensorSrcIIO::default_trigger
```

default set value of sampling frequency

Definition at line 131 of file gsttensor_srciio.h.

8.68.2.11 dev_dir

```
gchar* _GstTensorSrcIIO::dev_dir
```

Directory for device files

Definition at line 115 of file gsttensor_srciio.h.

8.68.2.12 device

```
GstTensorSrcIIODeviceProperties _GstTensorSrcIIO::device
```

IIO device

Definition at line 116 of file gsttensor_srciio.h.

8.68.2.13 element

```
GstBaseSrc _GstTensorSrcIIO::element
```

parent class object

Definition at line 106 of file gsttensor_srciio.h.

8.68.2.14 is_tensor

```
gboolean _GstTensorSrcIIO::is_tensor
```

False if tensors is used for data

Definition at line 125 of file gsttensor_srciio.h.

8.68.2.15 merge_channels_data

```
gboolean _GstTensorSrcIIO::merge_channels_data
```

merge channel data with same type/size

Definition at line 124 of file gsttensor_srciio.h.

8.68.2.16 mode

```
gchar* _GstTensorSrcIIO::mode
```

linux IIO related properties IIO device operating mode

Definition at line 113 of file gsttensor_srciio.h.

8.68.2.17 num_channels_enabled

```
guint _GstTensorSrcIIO::num_channels_enabled
```

channels to be enabled

Definition at line 123 of file gsttensor_srciio.h.

8.68.2.18 poll_timeout

```
gint _GstTensorSrcIIO::poll_timeout
```

timeout for polling the fifo file

Definition at line 132 of file gsttensor_srciio.h.

8.68.2.19 sampling_frequency

```
guint64 _GstTensorSrcIIO::sampling_frequency
```

sampling frequency for the device

Definition at line 127 of file gsttensor_srciio.h.

8.68.2.20 scan_size

```
guint _GstTensorSrcIIO::scan_size
```

size for a single scan of buffer length 1

Definition at line 121 of file gsttensor_srciio.h.

8.68.2.21 silent

```
gboolean _GstTensorSrcIIO::silent
```

gstreamer related properties true to print minimized log

Definition at line 109 of file gsttensor_srciio.h.

8.68.2.22 tensors_config

```
GstTensorsConfig* _GstTensorSrcIIO::tensors_config
```

Only first element is filled when is_tensor is true tensors for storing data config

Definition at line 135 of file gsttensor_srciio.h.

8.68.2.23 trigger

```
GstTensorSrcIIODeviceProperties _GstTensorSrcIIO::trigger
```

IIO trigger

Definition at line 117 of file gsttensor_srciio.h.

The documentation for this struct was generated from the following file:

- nnstreamer/elements/gsttensor_srciio.h

8.69 _GstTensorSrcIIOChannelProperties Struct Reference

GstTensorSrcIIO channel's properties (internal data structure)

```
#include <gsttensor_srciio.h>
```

Public Attributes

- gboolean enabled
- gboolean pre_enabled
- gchar * name
- gchar * generic_name
- gchar * base_dir
- gchar * base_file
- gint index
- gboolean big_endian
- gboolean is_signed
- guint used_bits
- guint64 mask
- guint storage_bytes
- guint storage_bits
- guint shift
- guint location
- gfloat offset
- gfloat scale

8.69.1 Detailed Description

GstTensorSrcIIO channel's properties (internal data structure)

Definition at line 77 of file gsttensor_srciio.h.

8.69.2 Member Data Documentation

8.69.2.1 base_dir

```
gchar* _GstTensorSrcIIOChannelProperties::base_dir
```

The base directory for the channel

Definition at line 83 of file gsttensor_srciio.h.

8.69.2.2 base_file

```
gchar* _GstTensorSrcIIOChannelProperties::base_file
```

The base filename for the channel

Definition at line 84 of file gsttensor_srciio.h.

8.69.2.3 big_endian

```
gboolean _GstTensorSrcIIOChannelProperties::big_endian
```

endian-ness of the data in buffer

Definition at line 87 of file gsttensor_srciio.h.

8.69.2.4 enabled

```
gboolean _GstTensorSrcIIOChannelProperties::enabled
```

currently state enabled/disabled

Definition at line 79 of file gsttensor_srciio.h.

8.69.2.5 generic_name

```
gchar* _GstTensorSrcIIOChannelProperties::generic_name
```

The generic name of the channel

Definition at line 82 of file gsttensor_srciio.h.

8.69.2.6 index

```
gint _GstTensorSrcIIOChannelProperties::index
```

index of the channel in the buffer

Definition at line 85 of file gsttensor_srciio.h.

8.69.2.7 is_signed

```
gboolean _GstTensorSrcIIOChannelProperties::is_signed
```

sign property of the data

Definition at line 88 of file gsttensor_srciio.h.

8.69.2.8 location

```
guint _GstTensorSrcIIOChannelProperties::location
```

location of channel data in buffer

Definition at line 94 of file gsttensor_srciio.h.

8.69.2.9 mask

```
guint64 _GstTensorSrcIIOChannelProperties::mask
```

size of the bits used for the data

Definition at line 90 of file gsttensor_srciio.h.

8.69.2.10 name

```
gchar* _GstTensorSrcIIOChannelProperties::name
```

The name of the channel

Definition at line 81 of file gsttensor_srciio.h.

8.69.2.11 offset

```
gfloat _GstTensorSrcIIOChannelProperties::offset
```

offset applied on raw data read from device

Definition at line 95 of file gsttensor_srciio.h.

8.69.2.12 pre_enabled

```
gboolean _GstTensorSrcIIOChannelProperties::pre_enabled
```

already in enabled/disabled state

Definition at line 80 of file gsttensor_srciio.h.

8.69.2.13 scale

```
gfloat _GstTensorSrcIIOChannelProperties::scale
```

scale applied on offset-ed data read from device

Definition at line 96 of file gsttensor_srciio.h.

8.69.2.14 shift

```
guint _GstTensorSrcIIOChannelProperties::shift
```

shift to be applied on the read data

Definition at line 93 of file gsttensor_srciio.h.

8.69.2.15 storage_bits

```
guint _GstTensorSrcIIOChannelProperties::storage_bits
```

exact bit size for the data

Definition at line 92 of file gsttensor_srciio.h.

8.69.2.16 storage_bytes

```
guint _GstTensorSrcIIOChannelProperties::storage_bytes
```

total storage size for the data

Definition at line 91 of file gsttensor_srciio.h.

8.69.2.17 used_bits

```
guint _GstTensorSrcIIOChannelProperties::used_bits
```

size of the bits used for the data

Definition at line 89 of file gsttensor_srciio.h.

The documentation for this struct was generated from the following file:

- nnstreamer/elements/gsttensor_srciio.h

8.70 _GstTensorSrcIIOClass Struct Reference

GstTensorSrcIIOClass data structure.

`#include <gsttensor_srciio.h>`

Public Attributes

- GstBaseSrcClass parent_class

8.70.1 Detailed Description

GstTensorSrcIIOClass data structure.

GstTensorSrcIIO inherits GstBaseSrc.

Definition at line 143 of file gsttensor_srciio.h.

8.70.2 Member Data Documentation

8.70.2.1 parent_class

`GstBaseSrcClass _GstTensorSrcIIOClass::parent_class`

inherits class object

Definition at line 145 of file gsttensor_srciio.h.

The documentation for this struct was generated from the following file:

- nnstreamer/elements/gsttensor_srciio.h

8.71 _GstTensorSrcIIODeviceProperties Struct Reference

GstTensorSrcIIO devices's properties (internal data structure)

`#include <gsttensor_srciio.h>`

Public Attributes

- gchar * name
- gchar * base_dir
- gint id

8.71.1 Detailed Description

GstTensorSrcIIO devices's properties (internal data structure)

This data structure is used for both device/triggers, as triggers are also iio devices

Definition at line 67 of file gsttensor_srciio.h.

8.71.2 Member Data Documentation

8.71.2.1 base_dir

```
gchar* _GstTensorSrcIIODeviceProperties::base_dir
```

The base directory for the device

Definition at line 70 of file gsttensor_srciio.h.

8.71.2.2 id

```
gint _GstTensorSrcIIODeviceProperties::id
```

The id of the device

Definition at line 71 of file gsttensor_srciio.h.

8.71.2.3 name

```
gchar* _GstTensorSrcIIODeviceProperties::name
```

The name of the device

Definition at line 69 of file gsttensor_srciio.h.

The documentation for this struct was generated from the following file:

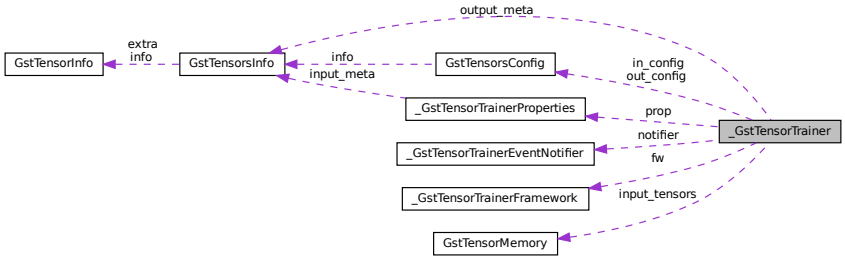
- nnstreamer/elements/gsttensor_srciio.h

8.72 _GstTensorTrainer Struct Reference

GstTensorTrainer data structure.

`#include <gsttensor_trainer.h>`

Collaboration diagram for _GstTensorTrainer:



Public Attributes

- GstElement element
- GstPad * sinkpad
- GstPad * srcpad
- gchar * fw_name
- gboolean fw_created
- gboolean is_training_complete
- gboolean is_epoch_complete
- GstTensorMemory input_tensors [NNS_TENSOR_SIZE_LIMIT]
- GstTensorsInfo output_meta
- GstTensorsConfig out_config
- GstTensorsConfig in_config
- guint required_sample
- guint cur_epoch_data_cnt
- void * privateData
- const GstTensorTrainerFramework * fw
- GstTensorTrainerProperties prop
- GstTensorTrainerEventNotifier notifier
- GMutex training_completion_lock
- GCond training_completion_cond
- GMutex epoch_completion_lock
- GCond epoch_completion_cond
- GThread * dummy_data_thread

8.72.1 Detailed Description

GstTensorTrainer data structure.

Definition at line 43 of file gsttensor_trainer.h.

8.72.2 Member Data Documentation

8.72.2.1 cur_epoch_data_cnt

```
guint _GstTensorTrainer::cur_epoch_data_cnt
```

number of total push data in one eposh

Definition at line 62 of file gsttensor_trainer.h.

8.72.2.2 dummy_data_thread

```
GThread* _GstTensorTrainer::dummy_data_thread
```

Dummy data generation thread

Definition at line 74 of file gsttensor_trainer.h.

8.72.2.3 element

```
GstElement _GstTensorTrainer::element
```

parent object

Definition at line 45 of file gsttensor_trainer.h.

8.72.2.4 epoch_completion_cond

```
GCond _GstTensorTrainer::epoch_completion_cond
```

Definition at line 72 of file gsttensor_trainer.h.

8.72.2.5 epoch_completion_lock

```
GMutex _GstTensorTrainer::epoch_completion_lock
```

Definition at line 71 of file gsttensor_trainer.h.

8.72.2.6 fw

```
const GstTensorTrainerFramework* _GstTensorTrainer::fw
```

Subplugin definition

Definition at line 65 of file gsttensor_trainer.h.

8.72.2.7 fw_created

```
gboolean _GstTensorTrainer::fw_created
```

Definition at line 52 of file gsttensor_trainer.h.

8.72.2.8 fw_name

```
gchar* _GstTensorTrainer::fw_name
```

Definition at line 50 of file gsttensor_trainer.h.

8.72.2.9 in_config

```
GstTensorsConfig _GstTensorTrainer::in_config
```

Definition at line 59 of file gsttensor_trainer.h.

8.72.2.10 input_tensors

```
GstTensorMemory _GstTensorTrainer::input_tensors[NNS_TENSOR_SIZE_LIMIT]
```

Definition at line 56 of file gsttensor_trainer.h.

8.72.2.11 is_epoch_complete

```
gboolean _GstTensorTrainer::is_epoch_complete
```

Definition at line 54 of file gsttensor_trainer.h.

8.72.2.12 is_training_complete

```
gboolean _GstTensorTrainer::is_training_complete
```

Definition at line 53 of file gsttensor_trainer.h.

8.72.2.13 notifier

```
GstTensorTrainerEventNotifier _GstTensorTrainer::notifier
```

Event notifier

Definition at line 67 of file gsttensor_trainer.h.

8.72.2.14 out_config

```
GstTensorsConfig _GstTensorTrainer::out_config
```

Definition at line 58 of file gsttensor_trainer.h.

8.72.2.15 output_meta

```
GstTensorsInfo _GstTensorTrainer::output_meta
```

Definition at line 57 of file gsttensor_trainer.h.

8.72.2.16 privateData

```
void* _GstTensorTrainer::privateData
```

NNFW plugin's private data is stored here

Definition at line 64 of file gsttensor_trainer.h.

8.72.2.17 prop

```
GstTensorTrainerProperties _GstTensorTrainer::prop
```

NNFW plugin's properties

Definition at line 66 of file gsttensor_trainer.h.

8.72.2.18 required_sample

```
guint _GstTensorTrainer::required_sample
```

Definition at line 61 of file gsttensor_trainer.h.

8.72.2.19 sinkpad

```
GstPad* _GstTensorTrainer::sinkpad
```

Definition at line 47 of file gsttensor_trainer.h.

8.72.2.20 srcpad

```
GstPad* _GstTensorTrainer::srcpad
```

Definition at line 48 of file gsttensor_trainer.h.

8.72.2.21 training_completion_cond

```
GCond _GstTensorTrainer::training_completion_cond
```

Definition at line 70 of file gsttensor_trainer.h.

8.72.2.22 training_completion_lock

```
GMutex _GstTensorTrainer::training_completion_lock
```

Definition at line 69 of file gsttensor_trainer.h.

The documentation for this struct was generated from the following file:

- nnstreamer/elements/gsttensor_trainer.h

8.73 _GstTensorTrainerClass Struct Reference

GstTensorTrainerClass data structure.

```
#include <gsttensor_trainer.h>
```

Public Attributes

- GstElementClass parent_class

8.73.1 Detailed Description

GstTensorTrainerClass data structure.

Definition at line 80 of file gsttensor_trainer.h.

8.73.2 Member Data Documentation

8.73.2.1 parent_class

```
GstElementClass _GstTensorTrainerClass::parent_class
```

parent class

Definition at line 82 of file gsttensor_trainer.h.

The documentation for this struct was generated from the following file:

- nnstreamer/elements/gsttensor_trainer.h

8.74 _GstTensorTrainerEventNotifier Struct Reference

GstTensorTrainer's event notifier.

```
#include <nnstreamer_plugin_api_trainer.h>
```

Public Attributes

- uint64_t version
- void * notifier

8.74.1 Detailed Description

GstTensorTrainer's event notifier.

Subplugins must send events to tensor_trainer with a notifier.

Definition at line 79 of file nnstreamer_plugin_api_trainer.h.

8.74.2 Member Data Documentation

8.74.2.1 notifier

```
void* _GstTensorTrainerEventNotifier::notifier
```

Version of the struct | 32bit (validity check) | 16bit (API version) | 16bit (Subplugin's internal version) |

Definition at line 82 of file nnstreamer_plugin_api_trainer.h.

8.74.2.2 version

```
uint64_t _GstTensorTrainerEventNotifier::version
```

Definition at line 81 of file nnstreamer_plugin_api_trainer.h.

The documentation for this struct was generated from the following file:

- nnstreamer/include/nnstreamer_plugin_api_trainer.h

8.75 _GstTensorTrainerFramework Struct Reference

tensor_trainer subplugin definition

```
#include <nnstreamer_plugin_api_trainer.h>
```

Public Attributes

- uint64_t version
- int(* create)(const GstTensorTrainerFramework *self, const GstTensorTrainerProperties *prop, void **private_data)
- int(* destroy)(const GstTensorTrainerFramework *self, const GstTensorTrainerProperties *prop, void **private_data)
- int(* start)(const GstTensorTrainerFramework *self, const GstTensorTrainerProperties *prop, GstTensorTrainerEventNotifier *notifier, void *private_data)
- int(* stop)(const GstTensorTrainerFramework *self, const GstTensorTrainerProperties *prop, void **private_data)
- int(* push_data)(const GstTensorTrainerFramework *self, const GstTensorTrainerProperties *prop, void *private_data, const GstTensorMemory *input)
- int(* getStatus)(const GstTensorTrainerFramework *self, GstTensorTrainerProperties *prop, void *private_data)
- int(* getFrameworkInfo)(const GstTensorTrainerFramework *self, const GstTensorTrainerProperties *prop, void *private_data, GstTensorTrainerFrameworkInfo *fw_info)

8.75.1 Detailed Description

tensor_trainer subplugin definition

Common callback parameters: prop Trainer properties. Read Only. private_data Subplugin's private data. Set this (∗private_data = XXX) if you want to change trainer->private_data.

Definition at line 95 of file nnstreamer_plugin_api_trainer.h.

8.75.2 Member Data Documentation

8.75.2.1 create

```
int(* _GstTensorTrainerFramework::create) (const GstTensorTrainerFramework *self, const GstTensorTrainerProper
*prop, void **private_data)
```

tensor_trainer call this to create the model

Parameters

in	*prop*	read-only property values
	[in/out]	private_data, a subplugin may save its internal private data here.

Returns

0 if ok. < 0 if error.

Definition at line 102 of file nnstreamer_plugin_api_trainer.h.

8.75.2.2 destroy

```
int(* _GstTensorTrainerFramework::destroy) (const GstTensorTrainerFramework *self, const GstTensorTrainerPrope
*prop, void **private_data)
```

tensor_trainer call this to destroy the model, Set NULL after that.

Parameters

in	*prop*	read-only property values
	[in/out]	private_data, a subplugin may save its internal private data here.

Returns

0 if ok. < 0 if error.

Definition at line 110 of file nnstreamer_plugin_api_trainer.h.

8.75.2.3 getFrameworkInfo

```
int(* _GstTensorTrainerFramework::getFrameworkInfo) (const GstTensorTrainerFramework *self,
const GstTensorTrainerProperties *prop, void *private_data, GstTensorTrainerFrameworkInfo *fw↩
_info)
```

Mandatory callback. Get the frameworks statically determined info.

Parameters

in	*prop*	read-only property values
in	*private_data*	A subplugin may save its internal private data here.
out	*fw_info*	struct to hold frameworks info. Must be allocated by the caller (return value).

Returns

0 if OK. < 0 if error.

Note

CAUTION: private_data can be NULL if the framework is not yet opened by the caller.

Definition at line 154 of file nnstreamer_plugin_api_trainer.h.

8.75.2.4 getStatus

```
int(* _GstTensorTrainerFramework::getStatus) (const GstTensorTrainerFramework *self, GstTensorTrainerPropertie
*prop, void *private_data)
```

tensor_trainer call this to get status of subplugin

Parameters

in	*prop*	property values
in	*private_data,a*	subplugin may save its internal private data here.

Returns

0 if ok. < 0 if error.

Definition at line 146 of file nnstreamer_plugin_api_trainer.h.

8.75.2.5 push_data

```
int(* _GstTensorTrainerFramework::push_data) (const GstTensorTrainerFramework *self, const
GstTensorTrainerProperties *prop, void *private_data, const GstTensorMemory *input)
```

tensor_trainer call this to push tensor data to subplugin, subplugin constructs a data set using input.

Parameters

in	*prop*	read-only property values
in	*private_data,a*	subplugin may save its internal private data here.
in	*input*	The array of input tensors. Allocated and filled by tensor_trainer

Returns

0 if ok. < 0 if error.

Definition at line 136 of file nnstreamer_plugin_api_trainer.h.

8.75.2.6 start

```
int(* _GstTensorTrainerFramework::start) (const GstTensorTrainerFramework *self, const GstTensorTrainerPropert
*prop, GstTensorTrainerEventNotifier *notifier, void *private_data)
```

tensor_trainer call this to start training the model

Parameters

in	*prop*	read-only property values
in	*notify,a*	handle of event notify
in	*private_data,a*	subplugin may save its internal private data here.

Returns

0 if ok. < 0 if error.

Definition at line 118 of file nnstreamer_plugin_api_trainer.h.

8.75.2.7 stop

```
int(* _GstTensorTrainerFramework::stop) (const GstTensorTrainerFramework *self, const GstTensorTrainerProperti
*prop, void **private_data)
```

tensor_trainer call this to stop training the model

Parameters

in	*prop*	read-only property values
	[in/out]	private_data, a subplugin may save its internal private data here.

Returns

0 if ok. < 0 if error.

Definition at line 128 of file nnstreamer_plugin_api_trainer.h.

8.75.2.8 version

`uint64_t _GstTensorTrainerFramework::version`

Version of the struct | 32bit (validity check) | 16bit (API version) | 16bit (Subplugin's internal version) |

Definition at line 97 of file nnstreamer_plugin_api_trainer.h.

The documentation for this struct was generated from the following file:

- nnstreamer/include/nnstreamer_plugin_api_trainer.h

8.76 _GstTensorTrainerFrameworkInfo Struct Reference

GstTensorTrainer's subplugin framework related information.

`#include <nnstreamer_plugin_api_trainer.h>`

Public Attributes

- const char * name

8.76.1 Detailed Description

GstTensorTrainer's subplugin framework related information.

All the information is provided statically.

Definition at line 55 of file nnstreamer_plugin_api_trainer.h.

8.76.2 Member Data Documentation

8.76.2.1 name

```
const char* _GstTensorTrainerFrameworkInfo::name
```

Name of the neural network framework, searchable by FRAMEWORK property.

Definition at line 57 of file nnstreamer_plugin_api_trainer.h.

The documentation for this struct was generated from the following file:

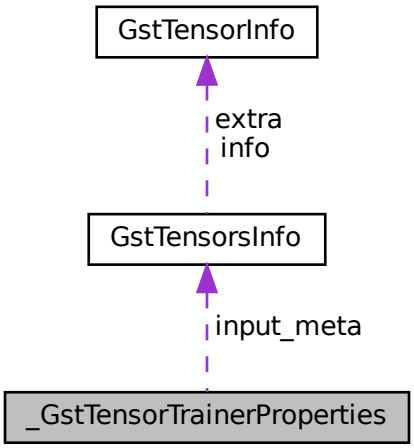
- nnstreamer/include/nnstreamer_plugin_api_trainer.h

8.77 _GstTensorTrainerProperties Struct Reference

GstTensorTrainer's properties for neural network framework (internal data structure)

```
#include <nnstreamer_plugin_api_trainer.h>
```

Collaboration diagram for _GstTensorTrainerProperties:



Public Attributes

- GstTensorsInfo input_meta
- const char * model_config
- const char * model_save_path
- const char * model_load_path
- unsigned int num_inputs
- unsigned int num_labels
- unsigned int num_training_samples
- unsigned int num_validation_samples
- unsigned int num_epochs
- unsigned int epoch_count
- double training_loss
- double training_accuracy
- double validation_loss
- double validation_accuracy

8.77.1 Detailed Description

GstTensorTrainer's properties for neural network framework (internal data structure)

Internal data of GstTensorTrainer required by tensor_trainer's custom subplugin.

Definition at line 31 of file nnstreamer_plugin_api_trainer.h.

8.77.2 Member Data Documentation

8.77.2.1 epoch_count

```
unsigned int _GstTensorTrainerProperties::epoch_count
```

Number of currently completed epochs

Definition at line 43 of file nnstreamer_plugin_api_trainer.h.

8.77.2.2 input_meta

```
GstTensorsInfo _GstTensorTrainerProperties::input_meta
```

configured input tensor info

Definition at line 33 of file nnstreamer_plugin_api_trainer.h.

8.77.2.3 model_config

```
const char* _GstTensorTrainerProperties::model_config
```

The configuration file path for creating model

Definition at line 34 of file nnstreamer_plugin_api_trainer.h.

8.77.2.4 model_load_path

```
const char* _GstTensorTrainerProperties::model_load_path
```

The file path to load an existing model to use for training a new model

Definition at line 36 of file nnstreamer_plugin_api_trainer.h.

8.77.2.5 model_save_path

```
const char* _GstTensorTrainerProperties::model_save_path
```

The file path to save the new model

Definition at line 35 of file nnstreamer_plugin_api_trainer.h.

8.77.2.6 num_epochs

```
unsigned int _GstTensorTrainerProperties::num_epochs
```

The number of repetition of total training and validation sample. subplugin must receive total samples((num_↩ training_samples + num_validation_samples) ∗ num_epochs)

Definition at line 41 of file nnstreamer_plugin_api_trainer.h.

8.77.2.7 num_inputs

```
unsigned int _GstTensorTrainerProperties::num_inputs
```

The number of input lists, the input is where framework receive the features to train the model, num_inputs indicates how many inputs there are.

Definition at line 37 of file nnstreamer_plugin_api_trainer.h.

8.77.2.8 num_labels

```
unsigned int _GstTensorTrainerProperties::num_labels
```

The number of label lists, the label is where framework receive the class to train the model, num_labels indicates how many labels there are.

Definition at line 38 of file nnstreamer_plugin_api_trainer.h.

8.77.2.9 num_training_samples

```
unsigned int _GstTensorTrainerProperties::num_training_samples
```

The number of training sample used to train the model.

Definition at line 39 of file nnstreamer_plugin_api_trainer.h.

8.77.2.10 num_validation_samples

```
unsigned int _GstTensorTrainerProperties::num_validation_samples
```

The number of validation sample used to valid the model.

Definition at line 40 of file nnstreamer_plugin_api_trainer.h.

8.77.2.11 training_accuracy

```
double _GstTensorTrainerProperties::training_accuracy
```

Training accuracy of the model being trained in the subplugin

Definition at line 45 of file nnstreamer_plugin_api_trainer.h.

8.77.2.12 training_loss

```
double _GstTensorTrainerProperties::training_loss
```

Training loss of the model being trained in the subplugin

Definition at line 44 of file nnstreamer_plugin_api_trainer.h.

8.77.2.13 validation_accuracy

```
double _GstTensorTrainerProperties::validation_accuracy
```

Validation accuracy of the model being trained in the subplugin

Definition at line 47 of file nnstreamer_plugin_api_trainer.h.

8.77.2.14 validation_loss

```
double _GstTensorTrainerProperties::validation_loss
```

Validation loss of the model being trained in the subplugin

Definition at line 46 of file nnstreamer_plugin_api_trainer.h.

The documentation for this struct was generated from the following file:

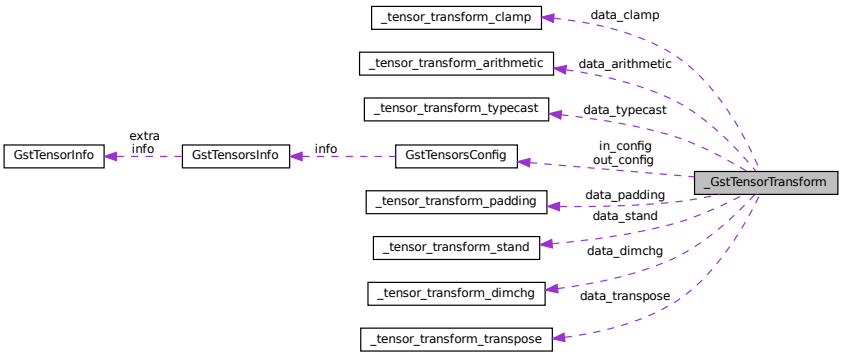
- nnstreamer/include/nnstreamer_plugin_api_trainer.h

8.78 _GstTensorTransform Struct Reference

Internal data structure for tensor_transform instances.

`#include <gsttensor_transform.h>`

Collaboration diagram for _GstTensorTransform:



Public Attributes

- GstBaseTransform element
- gboolean silent
- tensor_transform_mode mode
- gchar * option
- union {
  tensor_transform_dimchg data_dimchg
  tensor_transform_typecast data_typecast
  tensor_transform_arithmetic data_arithmetic
  tensor_transform_transpose data_transpose
  tensor_transform_stand data_stand
  tensor_transform_clamp data_clamp
  tensor_transform_padding data_padding
  };
- gboolean loaded
- gboolean acceleration
- GSList * operators
- GstTensorsConfig in_config
- GstTensorsConfig out_config
- GList * apply

8.78.1 Detailed Description

Internal data structure for tensor_transform instances.

Definition at line 166 of file gsttensor_transform.h.

8.78.2 Member Data Documentation

8.78.2.1 "@37

```
union { ... }
```

8.78.2.2 acceleration

```
gboolean _GstTensorTransform::acceleration
```

TRUE to set orc acceleration

Definition at line 183 of file gsttensor_transform.h.

8.78.2.3 apply

```
GList* _GstTensorTransform::apply
```

Select the tensors to apply transformation

Definition at line 188 of file gsttensor_transform.h.

8.78.2.4 data_arithmetic

```
tensor_transform_arithmetic _GstTensorTransform::data_arithmetic
```

Parsed option value for "arithmetic" mode.

Definition at line 176 of file gsttensor_transform.h.

8.78.2.5 data_clamp

```
tensor_transform_clamp _GstTensorTransform::data_clamp
```

Parsed option value for "clamp" mode.

Definition at line 179 of file gsttensor_transform.h.

8.78.2.6 data_dimchg

tensor_transform_dimchg _GstTensorTransform::data_dimchg

Parsed option value for "dimchg" mode

Definition at line 174 of file gsttensor_transform.h.

8.78.2.7 data_padding

tensor_transform_padding _GstTensorTransform::data_padding

Parsed option value for "padding" mode.

Definition at line 180 of file gsttensor_transform.h.

8.78.2.8 data_stand

tensor_transform_stand _GstTensorTransform::data_stand

Parsed option value for "stand" mode.

Definition at line 178 of file gsttensor_transform.h.

8.78.2.9 data_transpose

tensor_transform_transpose _GstTensorTransform::data_transpose

Parsed option value for "transpose" mode.

Definition at line 177 of file gsttensor_transform.h.

8.78.2.10 data_typecast

tensor_transform_typecast _GstTensorTransform::data_typecast

Parsed option value for "typecast" mode.

Definition at line 175 of file gsttensor_transform.h.

8.78.2.11 element

```
GstBaseTransform _GstTensorTransform::element
```

This is the parent object

Definition at line 168 of file gsttensor_transform.h.

8.78.2.12 in_config

```
GstTensorsConfig _GstTensorTransform::in_config
```

input tensors config

Definition at line 186 of file gsttensor_transform.h.

8.78.2.13 loaded

```
gboolean _GstTensorTransform::loaded
```

TRUE if mode & option are loaded

Definition at line 182 of file gsttensor_transform.h.

8.78.2.14 mode

```
tensor_transform_mode _GstTensorTransform::mode
```

Transform mode. GTT_UNKNOWN if invalid.

Definition at line 171 of file gsttensor_transform.h.

8.78.2.15 operators

```
GSList* _GstTensorTransform::operators
```

operators list

Definition at line 184 of file gsttensor_transform.h.

8.78.2.16 option

```
gchar* _GstTensorTransform::option
```

Stored option value

Definition at line 172 of file gsttensor_transform.h.

8.78.2.17 out_config

```
GstTensorsConfig _GstTensorTransform::out_config
```

output tensors config

Definition at line 187 of file gsttensor_transform.h.

8.78.2.18 silent

```
gboolean _GstTensorTransform::silent
```

True if logging is minimized

Definition at line 170 of file gsttensor_transform.h.

The documentation for this struct was generated from the following file:

- nnstreamer/elements/gsttensor_transform.h

8.79 _GstTensorTransformClass Struct Reference

GstTensorTransformClass inherits GstBaseTransformClass.

```
#include <gsttensor_transform.h>
```

Public Attributes

- GstBaseTransformClass parent_class

8.79.1 Detailed Description

GstTensorTransformClass inherits GstBaseTransformClass.

Referring another child (sibling), GstVideoFilter (abstract class) and its child (concrete class) GstVideoTransform. Note that GstTensorTransformClass is a concrete class; thus we need to look at both.

Definition at line 198 of file gsttensor_transform.h.

8.79.2 Member Data Documentation

8.79.2.1 parent_class

```
GstBaseTransformClass _GstTensorTransformClass::parent_class
```

Inherits GstBaseTransformClass

Definition at line 200 of file gsttensor_transform.h.

The documentation for this struct was generated from the following file:

- nnstreamer/elements/gsttensor_transform.h

8.80 _GTensorFilterSingle Struct Reference

Internal data structure for tensor_filter_single instances.

```
#include <tensor_filter_single.h>
```

Public Attributes

- GObject element
- gpointer priv

8.80.1 Detailed Description

Internal data structure for tensor_filter_single instances.

Definition at line 51 of file tensor_filter_single.h.

8.80.2 Member Data Documentation

8.80.2.1 element

```
GObject _GTensorFilterSingle::element
```

This is the parent object

Definition at line 53 of file tensor_filter_single.h.

8.80.2.2 priv

```
gpointer _GTensorFilterSingle::priv
```

Internal properties for tensor-filter single class

Definition at line 56 of file tensor_filter_single.h.

The documentation for this struct was generated from the following file:

- nnstreamer/tensor_filter/tensor_filter_single.h

8.81 _GTensorFilterSingleClass Struct Reference

GTensorFilterSingleClass inherits GObjectClass.

```
#include <tensor_filter_single.h>
```

Public Attributes

- GObjectClass parent
- gboolean(∗ invoke)(GTensorFilterSingle ∗self, const GstTensorMemory ∗input, GstTensorMemory ∗output, gboolean allocate)
- gboolean(∗ start)(GTensorFilterSingle ∗self)
- gboolean(∗ stop)(GTensorFilterSingle ∗self)
- gboolean(∗ input_configured)(GTensorFilterSingle ∗self)
- gboolean(∗ output_configured)(GTensorFilterSingle ∗self)
- gint(∗ set_input_info)(GTensorFilterSingle ∗self, const GstTensorsInfo ∗in_info, GstTensorsInfo ∗out_info)
- gboolean(∗ allocate_in_invoke)(GTensorFilterSingle ∗self)
- void(∗ destroy_notify)(GTensorFilterSingle ∗self, GstTensorMemory ∗mem)

8.81.1 Detailed Description

GTensorFilterSingleClass inherits GObjectClass.

Definition at line 62 of file tensor_filter_single.h.

8.81.2 Member Data Documentation

8.81.2.1 allocate_in_invoke

```
gboolean(* _GTensorFilterSingleClass::allocate_in_invoke) (GTensorFilterSingle *self)
```

Check if the filter performs allocate_in_invoke

Definition at line 81 of file tensor_filter_single.h.

8.81.2.2 destroy_notify

```
void(* _GTensorFilterSingleClass::destroy_notify) (GTensorFilterSingle *self, GstTensorMemory
*mem)
```

Free the data allocated by the tensor filter in invoke

Definition at line 83 of file tensor_filter_single.h.

8.81.2.3 input_configured

```
gboolean(* _GTensorFilterSingleClass::input_configured) (GTensorFilterSingle *self)
```

Check if the input is already configured

Definition at line 74 of file tensor_filter_single.h.

8.81.2.4 invoke

```
gboolean(* _GTensorFilterSingleClass::invoke) (GTensorFilterSingle *self, const GstTensorMemory
*input, GstTensorMemory *output, gboolean allocate)
```

Invoke the filter for execution.

Definition at line 67 of file tensor_filter_single.h.

8.81.2.5 output_configured

```
gboolean(* _GTensorFilterSingleClass::output_configured) (GTensorFilterSingle *self)
```

Check if the output is already configured

Definition at line 76 of file tensor_filter_single.h.

8.81.2.6 parent

```
GObjectClass _GTensorFilterSingleClass::parent
```

inherits GObjectClass

Definition at line 64 of file tensor_filter_single.h.

8.81.2.7 set_input_info

```
gint(* _GTensorFilterSingleClass::set_input_info) (GTensorFilterSingle *self, const GstTensorsInfo
*in_info, GstTensorsInfo *out_info)
```

Set the info about the input tensor

Definition at line 78 of file tensor_filter_single.h.

8.81.2.8 start

```
gboolean(* _GTensorFilterSingleClass::start) (GTensorFilterSingle *self)
```

Start the filter, must be called before invoke.

Definition at line 70 of file tensor_filter_single.h.

8.81.2.9 stop

```
gboolean(* _GTensorFilterSingleClass::stop) (GTensorFilterSingle *self)
```

Stop the filter.

Definition at line 72 of file tensor_filter_single.h.

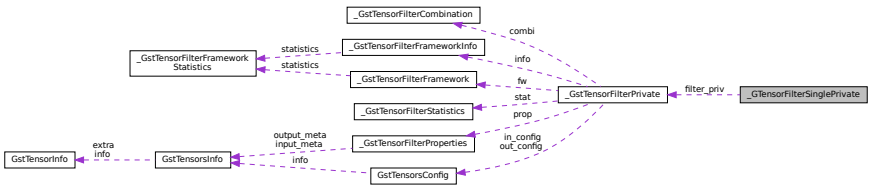
The documentation for this struct was generated from the following file:

- nnstreamer/tensor_filter/tensor_filter_single.h

8.82 _GTensorFilterSinglePrivate Struct Reference

Private data struct for tensor-filter single class.

Collaboration diagram for _GTensorFilterSinglePrivate:



Public Attributes

- GstTensorFilterPrivate filter_priv
- gboolean allocate_in_invoke

8.82.1 Detailed Description

Private data struct for tensor-filter single class.

SECTION:element-tensor_filter_single

An element that invokes neural network models and their framework or an independent shared object implementing tensor_filter_custom.h. The input and output are always in the format of other/tensor or other/tensors. This element is going to be the basis of single shot api.

Definition at line 53 of file tensor_filter_single.c.

8.82.2 Member Data Documentation

8.82.2.1 allocate_in_invoke

```
gboolean _GTensorFilterSinglePrivate::allocate_in_invoke
```

cached value after first invoke

Definition at line 56 of file tensor_filter_single.c.

8.82.2.2 filter_priv

```
GstTensorFilterPrivate _GTensorFilterSinglePrivate::filter_priv
```

Internal properties for tensor-filter

Definition at line 55 of file tensor_filter_single.c.

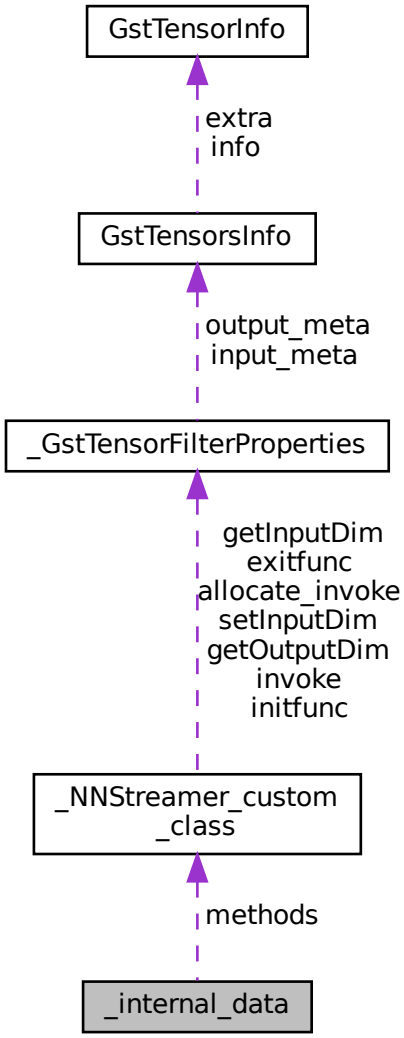
The documentation for this struct was generated from the following file:

- nnstreamer/tensor_filter/tensor_filter_single.c

8.83 _internal_data Struct Reference

internal_data

Collaboration diagram for _internal_data:



Public Attributes

- GModule * module
- NNStreamer_custom_class * methods
- void * customFW_private_data

8.83.1 Detailed Description

internal_data

Definition at line 50 of file tensor_filter_custom.c.

8.83.2 Member Data Documentation

8.83.2.1 customFW_private_data

```
void* _internal_data::customFW_private_data
```

Definition at line 55 of file tensor_filter_custom.c.

8.83.2.2 methods

```
NNStreamer_custom_class* _internal_data::methods
```

Definition at line 53 of file tensor_filter_custom.c.

8.83.2.3 module

```
GModule* _internal_data::module
```

Definition at line 52 of file tensor_filter_custom.c.

The documentation for this struct was generated from the following file:

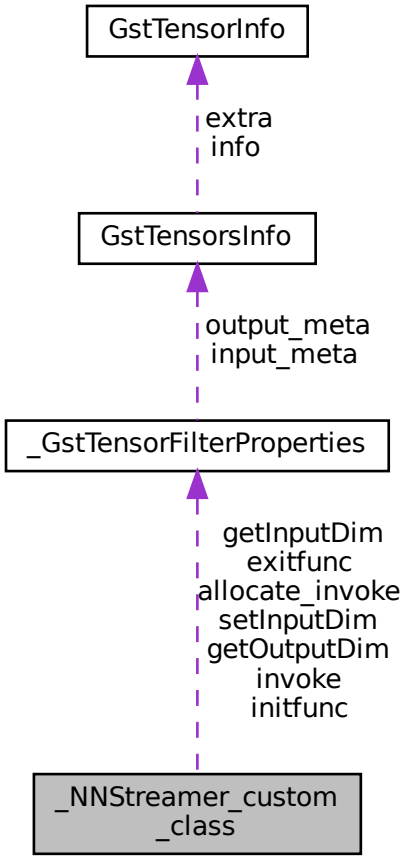
- nnstreamer/tensor_filter/tensor_filter_custom.c

8.84 _NNStreamer_custom_class Struct Reference

Custom Filter Class.

```
#include <tensor_filter_custom.h>
```

Collaboration diagram for _NNStreamer_custom_class:



Public Attributes

- NNS_custom_init_func initfunc
- NNS_custom_exit_func exitfunc
- NNS_custom_get_input_dimension getInputDim
- NNS_custom_get_output_dimension getOutputDim
- NNS_custom_set_input_dimension setInputDim
- NNS_custom_invoke invoke
- NNS_custom_allocate_invoke allocate_invoke
- NNS_custom_destroy_notify destroy_notify

8.84.1 Detailed Description

Custom Filter Class.

Note that every function pointer is MANDATORY!

Definition at line 125 of file tensor_filter_custom.h.

8.84.2 Member Data Documentation

8.84.2.1 allocate_invoke

NNS_custom_allocate_invoke _NNStreamer_custom_class::allocate_invoke

the main function, "allocate & invoke", that transforms input to output. allocate_invoke is supposed to allocate output buffer by itself. (invoke) XOR (allocate_invoke) MUST hold.

Definition at line 133 of file tensor_filter_custom.h.

8.84.2.2 destroy_notify

NNS_custom_destroy_notify _NNStreamer_custom_class::destroy_notify

it handles the data pointer allocated in the custom framework. when the data pointer has been destroyed at the pipeline, this method will be called. the data pointer or an object including data pointer could be deleted safely with this function. this method is only used when allocate_invoke is TRUE

Definition at line 134 of file tensor_filter_custom.h.

8.84.2.3 exitfunc

NNS_custom_exit_func _NNStreamer_custom_class::exitfunc

will not call other callbacks after this call

Definition at line 128 of file tensor_filter_custom.h.

8.84.2.4 getInputDim

NNS_custom_get_input_dimension _NNStreamer_custom_class::getInputDim

a custom filter is required to provide input tensor dimension unless setInputdim is defined.

Definition at line 129 of file tensor_filter_custom.h.

8.84.2.5 getOutputDim

NNS_custom_get_output_dimension _NNStreamer_custom_class::getOutputDim

a custom filter is required to provide output tensor dimension unless setInputDim is defined.

Definition at line 130 of file tensor_filter_custom.h.

8.84.2.6 initfunc

NNS_custom_init_func _NNStreamer_custom_class::initfunc

called before any other callbacks from tensor_filter_custom.c

Definition at line 127 of file tensor_filter_custom.h.

8.84.2.7 invoke

NNS_custom_invoke _NNStreamer_custom_class::invoke

the main function, "invoke", that transforms input to output. invoke is supposed to fill in the given output buffer. (invoke) XOR (allocate_invoke) MUST hold.

Definition at line 132 of file tensor_filter_custom.h.

8.84.2.8 setInputDim

NNS_custom_set_input_dimension _NNStreamer_custom_class::setInputDim

without getI/O-Dim, this allows framework to set input dimension and get output dimension from the custom filter according to the input dimension

Definition at line 131 of file tensor_filter_custom.h.

The documentation for this struct was generated from the following file:

- nnstreamer/include/tensor_filter_custom.h

8.85 _NNStreamerExternalConverter Struct Reference

Converter's subplugin implementation.

`#include <nnstreamer_plugin_api_converter.h>`

Public Attributes

- const char * name
- GstBuffer *(* convert)(GstBuffer *in_buf, GstTensorsConfig *config, void *priv_data)
- gboolean(* get_out_config)(const GstCaps *in_caps, GstTensorsConfig *config)
- GstCaps *(* query_caps)(const GstTensorsConfig *config)
- int(* open)(const gchar *script_path, void **priv_data)
- void(* close)(void **priv_data)

8.85.1 Detailed Description

Converter's subplugin implementation.

Definition at line 41 of file nnstreamer_plugin_api_converter.h.

8.85.2 Member Data Documentation

8.85.2.1 close

```
void(* _NNStreamerExternalConverter::close) (void **priv_data)
```

tensor_converter will call this to close subplugin.

Parameters

in	*private_data*	frees private_data and set NULL.

Definition at line 82 of file nnstreamer_plugin_api_converter.h.

8.85.2.2 convert

```
GstBuffer*(* _NNStreamerExternalConverter::convert) (GstBuffer *in_buf, GstTensorsConfig *config,
void *priv_data)
```

Convert the given input stream to tensor/tensors stream.

Parameters

in	*buf*	The input stream buffer
out	*config*	tensors config structure to be filled

Return values

Return	input buffer(in_buf) if the data is to be kept untouched.
Return	a new GstBuf if the data is to be modified.

Definition at line 48 of file nnstreamer_plugin_api_converter.h.

8.85.2.3 get_out_config

```
gboolean(* _NNStreamerExternalConverter::get_out_config) (const GstCaps *in_caps, GstTensorsConfig
*config)
```

Set the tensor config structure from the given stream frame.

Parameters

in	*in_caps*	The input (original/media data) stream's metadata
out	*config*	The output (tensor/tensors) metadata

Return values

Return	True if get caps successfully, FALSE if not.

Definition at line 58 of file nnstreamer_plugin_api_converter.h.

8.85.2.4 name

```
const char* _NNStreamerExternalConverter::name
```

Definition at line 45 of file nnstreamer_plugin_api_converter.h.

8.85.2.5 open

```
int(* _NNStreamerExternalConverter::open) (const gchar *script_path, void **priv_data)
```

tensor_converter will call this to open subplugin.

Parameters

in	*script_path*	script path of the subplugin.
	[in/out]	private_data A subplugin may save its internal private data here. The subplugin is responsible for alloc/free of this pointer. Normally, open() allocates memory for private_data.

Returns

0 if ok. < 0 if error.

Definition at line 75 of file nnstreamer_plugin_api_converter.h.

8.85.2.6 query_caps

```
GstCaps*(* _NNStreamerExternalConverter::query_caps) (const GstTensorsConfig *config)
```

Filters (narrows down) the GstCap (st) with the given config.

Parameters

in	*config*	The config of output tensor/tensors

Return values

Return	subplugin caps (if config is NULL, return default caps)

Definition at line 68 of file nnstreamer_plugin_api_converter.h.

The documentation for this struct was generated from the following file:

- nnstreamer/include/nnstreamer_plugin_api_converter.h

8.86 _NnstWatchdog Struct Reference

Structure for NNStreamer watchdog.

Public Attributes

- GMainContext * context
- GMainLoop * loop
- GThread * thread
- GSource * source
- GMutex lock
- GCond cond

8.86.1 Detailed Description

Structure for NNStreamer watchdog.

Definition at line 21 of file nnstreamer_watchdog.c.

8.86.2 Member Data Documentation

8.86.2.1 cond

```
GCond _NnstWatchdog::cond
```

Definition at line 28 of file nnstreamer_watchdog.c.

8.86.2.2 context

```
GMainContext* _NnstWatchdog::context
```

Definition at line 23 of file nnstreamer_watchdog.c.

8.86.2.3 lock

```
GMutex _NnstWatchdog::lock
```

Definition at line 27 of file nnstreamer_watchdog.c.

8.86.2.4 loop

```
GMainLoop* _NnstWatchdog::loop
```

Definition at line 24 of file nnstreamer_watchdog.c.

8.86.2.5 source

```
GSource* _NnstWatchdog::source
```

Definition at line 26 of file nnstreamer_watchdog.c.

8.86.2.6 thread

```
GThread* _NnstWatchdog::thread
```

Definition at line 25 of file nnstreamer_watchdog.c.

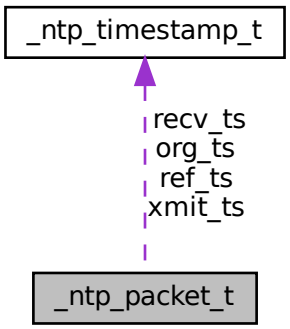
The documentation for this struct was generated from the following file:

- nnstreamer/nnstreamer_watchdog.c

8.87 _ntp_packet_t Struct Reference

A custom data type to represent NTP packet header format.

Collaboration diagram for _ntp_packet_t:



Public Attributes

- uint8_t li_vn_mode
- uint8_t stratum
- uint8_t poll
- uint8_t precision
- uint32_t root_delay
- uint32_t root_dispersion
- uint32_t ref_id
- ntp_timestamp_t ref_ts
- ntp_timestamp_t org_ts
- ntp_timestamp_t recv_ts
- ntp_timestamp_t xmit_ts

8.87.1 Detailed Description

A custom data type to represent NTP packet header format.

NTP Packet Header Format (https://www.ietf.org/rfc/rfc5905.txt p.18) 0 1 2 3 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 +-+ |LI | VN |Mode | Stratum | Poll | Precision | +-+ | Root Delay | +-+ | Root Dispersion | +-+ | Reference ID | +-+ | |

- Reference Timestamp (64) + | | +-+ | |
- Origin Timestamp (64) + | | +-+ | |
- Receive Timestamp (64) + | | +-+ | |
- Transmit Timestamp (64) + | | +-+ | | . . . Extension Field 1 (variable) . . . | | +-+ | | . . . Extension Field 2 (variable) . . . | | +-+ | Key Identifier | +-+ | | dgst (128) | | | +-+

Definition at line 100 of file ntputil.c.

8.87.2 Member Data Documentation

8.87.2.1 li_vn_mode

```
uint8_t _ntp_packet_t::li_vn_mode
```

Definition at line 102 of file ntputil.c.

8.87.2.2 org_ts

```
ntp_timestamp_t _ntp_packet_t::org_ts
```

Definition at line 110 of file ntputil.c.

8.87.2.3 poll

```
uint8_t _ntp_packet_t::poll
```

Definition at line 104 of file ntputil.c.

8.87.2.4 precision

```
uint8_t _ntp_packet_t::precision
```

Definition at line 105 of file ntputil.c.

8.87.2.5 recv_ts

```
ntp_timestamp_t _ntp_packet_t::recv_ts
```

Definition at line 111 of file ntputil.c.

8.87.2.6 ref_id

```
uint32_t _ntp_packet_t::ref_id
```

Definition at line 108 of file ntputil.c.

8.87.2.7 ref_ts

```
ntp_timestamp_t _ntp_packet_t::ref_ts
```

Definition at line 109 of file ntputil.c.

8.87.2.8 root_delay

```
uint32_t _ntp_packet_t::root_delay
```

Definition at line 106 of file ntputil.c.

8.87.2.9 root_dispersion

```
uint32_t _ntp_packet_t::root_dispersion
```

Definition at line 107 of file ntputil.c.

8.87.2.10 stratum

```
uint8_t _ntp_packet_t::stratum
```

Definition at line 103 of file ntputil.c.

8.87.2.11 xmit_ts

```
ntp_timestamp_t _ntp_packet_t::xmit_ts
```

Definition at line 112 of file ntputil.c.

The documentation for this struct was generated from the following file:

- mqtt/ntputil.c

8.88 _ntp_timestamp_t Struct Reference

A custom data type to represent NTP timestamp format.

Public Attributes

- uint32_t sec
- uint32_t frac

8.88.1 Detailed Description

A custom data type to represent NTP timestamp format.

NTP Timestamp Format (https://www.ietf.org/rfc/rfc5905.txt p.12) 0 1 2 3 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 +-+ | Seconds | +-+ | Fraction | +-+

Definition at line 40 of file ntputil.c.

8.88.2 Member Data Documentation

8.88.2.1 frac

```
uint32_t _ntp_timestamp_t::frac
```

Definition at line 43 of file ntputil.c.

8.88.2.2 sec

```
uint32_t _ntp_timestamp_t::sec
```

Definition at line 42 of file ntputil.c.

The documentation for this struct was generated from the following file:

- mqtt/ntputil.c

8.89 _tensor_merge_linear Struct Reference

Internal data structure for linear mode.

```
#include <gsttensor_merge.h>
```

Public Attributes

- tensor_merge_linear_mode direction

8.89.1 Detailed Description

Internal data structure for linear mode.

Definition at line 64 of file gsttensor_merge.h.

8.89.2 Member Data Documentation

8.89.2.1 direction

```
tensor_merge_linear_mode _tensor_merge_linear::direction
```

Definition at line 65 of file gsttensor_merge.h.

The documentation for this struct was generated from the following file:

- nnstreamer/elements/gsttensor_merge.h

8.90 _tensor_sync_basepad_data Struct Reference

Tensor Merge/Mux sync data for baspad mode.

```
#include <tensor_common.h>
```

Public Attributes

- guint sink_id
- GstClockTime duration

8.90.1 Detailed Description

Tensor Merge/Mux sync data for baspad mode.

Definition at line 74 of file tensor_common.h.

8.90.2 Member Data Documentation

8.90.2.1 duration

```
GstClockTime _tensor_sync_basepad_data::duration
```

Definition at line 76 of file tensor_common.h.

8.90.2.2 sink_id

```
guint _tensor_sync_basepad_data::sink_id
```

Definition at line 75 of file tensor_common.h.

The documentation for this struct was generated from the following file:

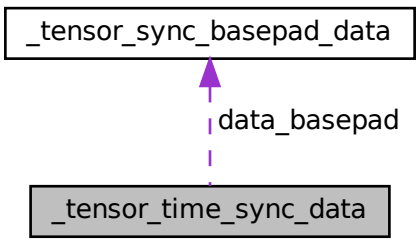
- nnstreamer/tensor_common.h

8.91 _tensor_time_sync_data Struct Reference

Tensor Merge/Mux time sync data.

```
#include <tensor_common.h>
```

Collaboration diagram for _tensor_time_sync_data:



Public Attributes

- tensor_time_sync_mode mode
- gchar * option
- union {
  tensor_sync_basepad_data data_basepad
  };

8.91.1 Detailed Description

Tensor Merge/Mux time sync data.

Definition at line 82 of file tensor_common.h.

8.91.2 Member Data Documentation

8.91.2.1 "@61

```
union { ... }
```

8.91.2.2 data_basepad

```
tensor_sync_basepad_data _tensor_time_sync_data::data_basepad
```

Definition at line 86 of file tensor_common.h.

8.91.2.3 mode

```
tensor_time_sync_mode _tensor_time_sync_data::mode
```

Definition at line 83 of file tensor_common.h.

8.91.2.4 option

```
gchar* _tensor_time_sync_data::option
```

Definition at line 84 of file tensor_common.h.

The documentation for this struct was generated from the following file:

- nnstreamer/tensor_common.h

8.92 _tensor_transform_arithmetic Struct Reference

Internal data structure for arithmetic mode.

`#include <gsttensor_transform.h>`

Public Attributes

- tensor_type out_type
- gboolean per_channel_arith
- guint ch_dim

8.92.1 Detailed Description

Internal data structure for arithmetic mode.

Definition at line 126 of file gsttensor_transform.h.

8.92.2 Member Data Documentation

8.92.2.1 ch_dim

`guint _tensor_transform_arithmetic::ch_dim`

Definition at line 129 of file gsttensor_transform.h.

8.92.2.2 out_type

`tensor_type _tensor_transform_arithmetic::out_type`

Definition at line 127 of file gsttensor_transform.h.

8.92.2.3 per_channel_arith

`gboolean _tensor_transform_arithmetic::per_channel_arith`

Definition at line 128 of file gsttensor_transform.h.

The documentation for this struct was generated from the following file:

- nnstreamer/elements/gsttensor_transform.h

8.93 _tensor_transform_clamp Struct Reference

Internal data structure for clamp mode.

`#include <gsttensor_transform.h>`

Public Attributes

- double min
- double max

8.93.1 Detailed Description

Internal data structure for clamp mode.

Definition at line 151 of file gsttensor_transform.h.

8.93.2 Member Data Documentation

8.93.2.1 max

`double _tensor_transform_clamp::max`

Definition at line 152 of file gsttensor_transform.h.

8.93.2.2 min

`double _tensor_transform_clamp::min`

Definition at line 152 of file gsttensor_transform.h.

The documentation for this struct was generated from the following file:

- nnstreamer/elements/gsttensor_transform.h

8.94 _tensor_transform_dimchg Struct Reference

Internal data structure for dimchg mode.

`#include <gsttensor_transform.h>`

Public Attributes

- int from
- int to

8.94.1 Detailed Description

Internal data structure for dimchg mode.

Definition at line 101 of file gsttensor_transform.h.

8.94.2 Member Data Documentation

8.94.2.1 from

```
int _tensor_transform_dimchg::from
```

Definition at line 102 of file gsttensor_transform.h.

8.94.2.2 to

```
int _tensor_transform_dimchg::to
```

Definition at line 103 of file gsttensor_transform.h.

The documentation for this struct was generated from the following file:

- nnstreamer/elements/gsttensor_transform.h

8.95 _tensor_transform_padding Struct Reference

Internal data structure for padding mode.

```
#include <gsttensor_transform.h>
```

Public Attributes

- guint pad [NNS_TENSOR_RANK_LIMIT]
- tensor_layout layout

8.95.1 Detailed Description

Internal data structure for padding mode.

Definition at line 158 of file gsttensor_transform.h.

8.95.2 Member Data Documentation

8.95.2.1 layout

```
tensor_layout _tensor_transform_padding::layout
```

Definition at line 160 of file gsttensor_transform.h.

8.95.2.2 pad

```
guint _tensor_transform_padding::pad[NNS_TENSOR_RANK_LIMIT]
```

Definition at line 159 of file gsttensor_transform.h.

The documentation for this struct was generated from the following file:

- nnstreamer/elements/gsttensor_transform.h

8.96 _tensor_transform_stand Struct Reference

Internal data structure for stand mode.

```
#include <gsttensor_transform.h>
```

Public Attributes

- tensor_transform_stand_mode mode
- tensor_type out_type
- gboolean per_channel

8.96.1 Detailed Description

Internal data structure for stand mode.

Definition at line 142 of file gsttensor_transform.h.

8.96.2 Member Data Documentation

8.96.2.1 mode

```
tensor_transform_stand_mode _tensor_transform_stand::mode
```

Definition at line 143 of file gsttensor_transform.h.

8.96.2.2 out_type

```
tensor_type _tensor_transform_stand::out_type
```

Definition at line 144 of file gsttensor_transform.h.

8.96.2.3 per_channel

```
gboolean _tensor_transform_stand::per_channel
```

Definition at line 145 of file gsttensor_transform.h.

The documentation for this struct was generated from the following file:

- nnstreamer/elements/gsttensor_transform.h

8.97 _tensor_transform_transpose Struct Reference

Internal data structure for transpose mode.

```
#include <gsttensor_transform.h>
```

Public Attributes

- uint8_t trans_order [NNS_TENSOR_RANK_LIMIT]

8.97.1 Detailed Description

Internal data structure for transpose mode.

Definition at line 135 of file gsttensor_transform.h.

8.97.2 Member Data Documentation

8.97.2.1 trans_order

```
uint8_t _tensor_transform_transpose::trans_order[NNS_TENSOR_RANK_LIMIT]
```

Definition at line 136 of file gsttensor_transform.h.

The documentation for this struct was generated from the following file:

- nnstreamer/elements/gsttensor_transform.h

8.98 _tensor_transform_typecast Struct Reference

Internal data structure for typecast mode.

```
#include <gsttensor_transform.h>
```

Public Attributes

- tensor_type to

8.98.1 Detailed Description

Internal data structure for typecast mode.

Definition at line 109 of file gsttensor_transform.h.

8.98.2 Member Data Documentation

8.98.2.1 to

```
tensor_type _tensor_transform_typecast::to
```

tensor_type after cast. _NNS_END if unknown

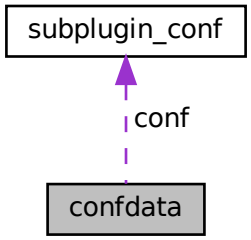
Definition at line 110 of file gsttensor_transform.h.

The documentation for this struct was generated from the following file:

- nnstreamer/elements/gsttensor_transform.h

8.99 confdata Struct Reference

Collaboration diagram for confdata:



Public Attributes

- gboolean loaded
- gboolean enable_envvar
- gboolean enable_symlink
- gchar * conffile
- gchar * extra_conffile
- subplugin_conf conf [NNSCONF_PATH_END]

8.99.1 Detailed Description

Definition at line 97 of file nnstreamer_conf.c.

8.99.2 Member Data Documentation

8.99.2.1 conf

```
subplugin_conf confdata::conf[NNSCONF_PATH_END]
```

Definition at line 106 of file nnstreamer_conf.c.

8.99.2.2 conffile

```
gchar* confdata::conffile
```

Location of conf file.

Definition at line 103 of file nnstreamer_conf.c.

8.99.2.3 enable_envvar

```
gboolean confdata::enable_envvar
```

TRUE to parse env variables

Definition at line 100 of file nnstreamer_conf.c.

8.99.2.4 enable_symlink

```
gboolean confdata::enable_symlink
```

TRUE to allow symbolic link file

Definition at line 101 of file nnstreamer_conf.c.

8.99.2.5 extra_conffile

```
gchar* confdata::extra_conffile
```

Location of extra configuration file.

Definition at line 104 of file nnstreamer_conf.c.

8.99.2.6 loaded

```
gboolean confdata::loaded
```

TRUE if loaded at least once

Definition at line 99 of file nnstreamer_conf.c.

The documentation for this struct was generated from the following file:

- nnstreamer/nnstreamer_conf.c

8.100 converter_custom_cb_s Struct Reference

`#include <gsttensor_converter.h>`

Public Attributes

- tensor_converter_custom func
- void * data

8.100.1 Detailed Description

Definition at line 57 of file gsttensor_converter.h.

8.100.2 Member Data Documentation

8.100.2.1 data

```
void* converter_custom_cb_s::data
```

Definition at line 60 of file gsttensor_converter.h.

8.100.2.2 func

```
tensor_converter_custom converter_custom_cb_s::func
```

Definition at line 59 of file gsttensor_converter.h.

The documentation for this struct was generated from the following file:

- nnstreamer/elements/gsttensor_converter.h

8.101 custom_cb_s Struct Reference

`#include <gsttensor_if.h>`

Public Attributes

- gchar * name
- tensor_if_custom func
- void * data

8.101.1 Detailed Description

Definition at line 112 of file gsttensor_if.h.

8.101.2 Member Data Documentation

8.101.2.1 data

```
void* custom_cb_s::data
```

Definition at line 116 of file gsttensor_if.h.

8.101.2.2 func

```
tensor_if_custom custom_cb_s::func
```

Definition at line 115 of file gsttensor_if.h.

8.101.2.3 name

```
gchar* custom_cb_s::name
```

Definition at line 114 of file gsttensor_if.h.

The documentation for this struct was generated from the following file:

- nnstreamer/elements/gsttensor_if.h

8.102 decoder_custom_cb_s Struct Reference

```
#include <gsttensor_decoder.h>
```

Public Attributes

- tensor_decoder_custom func
- void * data

8.102.1 Detailed Description

Definition at line 56 of file gsttensor_decoder.h.

8.102.2 Member Data Documentation

8.102.2.1 data

```
void* decoder_custom_cb_s::data
```

Definition at line 59 of file gsttensor_decoder.h.

8.102.2.2 func

```
tensor_decoder_custom decoder_custom_cb_s::func
```

Definition at line 58 of file gsttensor_decoder.h.

The documentation for this struct was generated from the following file:

- nnstreamer/elements/gsttensor_decoder.h

8.103 dump_buf Struct Reference

Public Attributes

- gchar * base
- gulong size
- gulong pos

8.103.1 Detailed Description

Definition at line 670 of file nnstreamer_conf.c.

8.103.2 Member Data Documentation

8.103.2.1 base

```
gchar* dump_buf::base
```

Definition at line 672 of file nnstreamer_conf.c.

8.103.2.2 pos

```
gulong dump_buf::pos
```

Definition at line 674 of file nnstreamer_conf.c.

8.103.2.3 size

```
gulong dump_buf::size
```

Definition at line 673 of file nnstreamer_conf.c.

The documentation for this struct was generated from the following file:

- nnstreamer/nnstreamer_conf.c

8.104 gst_tensor_aggregation_data_s Struct Reference

Internal struct to handle aggregation data in hash table.

Public Attributes

- GstAdapter * adapter

8.104.1 Detailed Description

Internal struct to handle aggregation data in hash table.

Definition at line 657 of file nnstreamer_plugin_api_impl.c.

8.104.2 Member Data Documentation

8.104.2.1 adapter

```
GstAdapter* gst_tensor_aggregation_data_s::adapter
```

Definition at line 659 of file nnstreamer_plugin_api_impl.c.

The documentation for this struct was generated from the following file:

- nnstreamer/nnstreamer_plugin_api_impl.c

8.105 GstMetaQuery Struct Reference

GstMetaQuery meta structure.

```
#include <tensor_meta.h>
```

Public Attributes

- GstMeta meta
- query_client_id_t client_id

8.105.1 Detailed Description

GstMetaQuery meta structure.

Definition at line 26 of file tensor_meta.h.

8.105.2 Member Data Documentation

8.105.2.1 client_id

```
query_client_id_t GstMetaQuery::client_id
```

Definition at line 30 of file tensor_meta.h.

8.105.2.2 meta

```
GstMeta GstMetaQuery::meta
```

Definition at line 28 of file tensor_meta.h.

The documentation for this struct was generated from the following file:

- nnstreamer/tensor_meta.h

8.106 GstSparseTensorInfo Struct Reference

Internal data structure for sparse tensor info.

`#include <tensor_typedef.h>`

Public Attributes

- uint32_t nnz

8.106.1 Detailed Description

Internal data structure for sparse tensor info.

Definition at line 294 of file tensor_typedef.h.

8.106.2 Member Data Documentation

8.106.2.1 nnz

`uint32_t GstSparseTensorInfo::nnz`

the number of "non-zero" elements

Definition at line 296 of file tensor_typedef.h.

The documentation for this struct was generated from the following file:

- nnstreamer/include/tensor_typedef.h

8.107 GstTensorAllocator Struct Reference

struct for type GstTensorAllocator

Public Attributes

- GstAllocator parent

8.107.1 Detailed Description

struct for type GstTensorAllocator

Definition at line 24 of file tensor_allocator.c.

8.107.2 Member Data Documentation

8.107.2.1 parent

```
GstAllocator GstTensorAllocator::parent
```

Definition at line 26 of file tensor_allocator.c.

The documentation for this struct was generated from the following file:

- nnstreamer/tensor_allocator.c

8.108 GstTensorAllocatorClass Struct Reference

struct for class GstTensorAllocatorClass

Public Attributes

- GstAllocatorClass parent_class

8.108.1 Detailed Description

struct for class GstTensorAllocatorClass

Definition at line 32 of file tensor_allocator.c.

8.108.2 Member Data Documentation

8.108.2.1 parent_class

```
GstAllocatorClass GstTensorAllocatorClass::parent_class
```

Definition at line 34 of file tensor_allocator.c.

The documentation for this struct was generated from the following file:

- nnstreamer/tensor_allocator.c

8.109 GstTensorCollectPadData Struct Reference

Internal data structure for Collect Pad in mux / merge.

`#include <tensor_common.h>`

Public Attributes

- GstCollectData collect
- GstBuffer * buffer

8.109.1 Detailed Description

Internal data structure for Collect Pad in mux / merge.

Definition at line 93 of file tensor_common.h.

8.109.2 Member Data Documentation

8.109.2.1 buffer

`GstBuffer* GstTensorCollectPadData::buffer`

Definition at line 96 of file tensor_common.h.

8.109.2.2 collect

`GstCollectData GstTensorCollectPadData::collect`

Definition at line 95 of file tensor_common.h.

The documentation for this struct was generated from the following file:

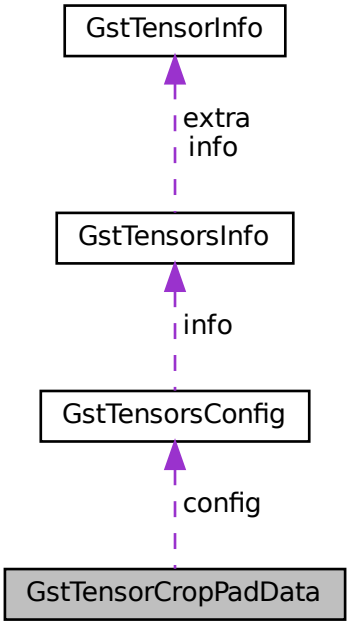
- nnstreamer/tensor_common.h

8.110 GstTensorCropPadData Struct Reference

GstTensorCrop pad data.

`#include <gsttensor_crop.h>`

Collaboration diagram for GstTensorCropPadData:



Public Attributes

- GstCollectData data
- GstTensorsConfig config

8.110.1 Detailed Description

GstTensorCrop pad data.

Definition at line 39 of file gsttensor_crop.h.

8.110.2 Member Data Documentation

8.110.2.1 config

```
GstTensorsConfig GstTensorCropPadData::config
```

Definition at line 43 of file gsttensor_crop.h.

8.110.2.2 data

```
GstCollectData GstTensorCropPadData::data
```

Definition at line 41 of file gsttensor_crop.h.

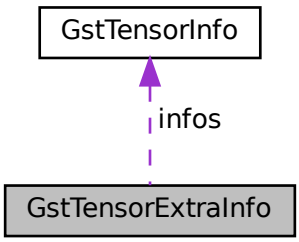
The documentation for this struct was generated from the following file:

- nnstreamer/elements/gsttensor_crop.h

8.111 GstTensorExtraInfo Struct Reference

Data structure to describe a "extra" tensor data. This represents the information of the NNS_TENSOR_SIZE_LI↩ MIT-th memory block for tensor stream.

Collaboration diagram for GstTensorExtraInfo:



Public Attributes

- uint32_t magic
- uint32_t version
- uint32_t num_extra_tensors
- uint64_t reserved
- GstTensorInfo infos [NNS_TENSOR_SIZE_EXTRA_LIMIT]

8.111.1 Detailed Description

Data structure to describe a "extra" tensor data. This represents the information of the NNS_TENSOR_SIZE_LI↩ MIT-th memory block for tensor stream.

Definition at line 39 of file nnstreamer_plugin_api_impl.c.

8.111.2 Member Data Documentation

8.111.2.1 infos

```
GstTensorInfo GstTensorExtraInfo::infos[NNS_TENSOR_SIZE_EXTRA_LIMIT]
```

Definition at line 45 of file nnstreamer_plugin_api_impl.c.

8.111.2.2 magic

```
uint32_t GstTensorExtraInfo::magic
```

Definition at line 41 of file nnstreamer_plugin_api_impl.c.

8.111.2.3 num_extra_tensors

```
uint32_t GstTensorExtraInfo::num_extra_tensors
```

Definition at line 43 of file nnstreamer_plugin_api_impl.c.

8.111.2.4 reserved

```
uint64_t GstTensorExtraInfo::reserved
```

Definition at line 44 of file nnstreamer_plugin_api_impl.c.

8.111.2.5 version

```
uint32_t GstTensorExtraInfo::version
```

Definition at line 42 of file nnstreamer_plugin_api_impl.c.

The documentation for this struct was generated from the following file:

- nnstreamer/nnstreamer_plugin_api_impl.c

8.112 GstTensorFilterSharedModelRepresentation Struct Reference

Data Structure to store shared table.

```
#include <tensor_filter_common.h>
```

Public Attributes

- void * shared_interpreter
- GList * referred_list

8.112.1 Detailed Description

Data Structure to store shared table.

Definition at line 144 of file tensor_filter_common.h.

8.112.2 Member Data Documentation

8.112.2.1 referred_list

```
GList* GstTensorFilterSharedModelRepresentation::referred_list
```

the referred list about the instances sharing the same key

Definition at line 146 of file tensor_filter_common.h.

8.112.2.2 shared_interpreter

```
void* GstTensorFilterSharedModelRepresentation::shared_interpreter
```

the model representation for each sub-plugins

Definition at line 145 of file tensor_filter_common.h.

The documentation for this struct was generated from the following file:

- nnstreamer/tensor_filter/tensor_filter_common.h

8.113 GstTensorInfo Struct Reference

Internal data structure for tensor info.

```
#include <tensor_typedef.h>
```

Public Attributes

- char * name
- tensor_type type
- tensor_dim dimension

8.113.1 Detailed Description

Internal data structure for tensor info.

Definition at line 261 of file tensor_typedef.h.

8.113.2 Member Data Documentation

8.113.2.1 dimension

```
tensor_dim GstTensorInfo::dimension
```

Dimension. We support up to 16th ranks.

Definition at line 267 of file tensor_typedef.h.

8.113.2.2 name

```
char* GstTensorInfo::name
```

Name of each element in the tensor. User must designate this in a few NNFW frameworks (tensorflow) and some (tensorflow-lite) do not need this.

Definition at line 263 of file tensor_typedef.h.

8.113.2.3 type

```
tensor_type GstTensorInfo::type
```

Type of each element in the tensor. User must designate this.

Definition at line 266 of file tensor_typedef.h.

The documentation for this struct was generated from the following file:

- nnstreamer/include/tensor_typedef.h

8.114 GstTensorMemory Struct Reference

The unit of each data tensors. It will be used as an input/output tensor of other/tensors.

```
#include <tensor_typedef.h>
```

Public Attributes

- void * data
- size_t size

8.114.1 Detailed Description

The unit of each data tensors. It will be used as an input/output tensor of other/tensors.

Definition at line 252 of file tensor_typedef.h.

8.114.2 Member Data Documentation

8.114.2.1 data

```
void* GstTensorMemory::data
```

The instance of tensor data.

Definition at line 254 of file tensor_typedef.h.

8.114.2.2 size

```
size_t GstTensorMemory::size
```

The size of tensor.

Definition at line 255 of file tensor_typedef.h.

The documentation for this struct was generated from the following file:

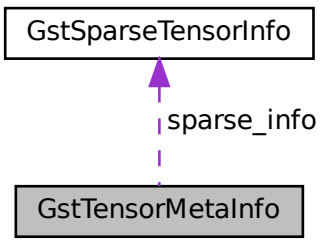
- nnstreamer/include/tensor_typedef.h

8.115 GstTensorMetaInfo Struct Reference

Data structure to describe a tensor data. This represents the basic information of a memory block for tensor stream.

```
#include <tensor_typedef.h>
```

Collaboration diagram for GstTensorMetaInfo:



Public Attributes

- uint32_t magic
- uint32_t version
- uint32_t type
- tensor_dim dimension
- uint32_t format
- uint32_t media_type
- union {
  GstSparseTensorInfo sparse_info
  };

  Union of the required information for processing each tensor "format".

8.115.1 Detailed Description

Data structure to describe a tensor data. This represents the basic information of a memory block for tensor stream.

Internally NNStreamer handles a buffer with capability other/tensors-flexible using this information.

- version: The version of tensor meta.
- type: The type of each element in the tensor. This should be a value of enumeration tensor_type.
- dimension: The dimension of tensor. This also denotes the rank of tensor. (e.g., [3:224:224:0] means rank 3.)
- format: The data format in the tensor. This should be a value of enumeration tensor_format.
- media_type: The media type of tensor. This should be a value of enumeration media_type.

Definition at line 310 of file tensor_typedef.h.

8.115.2 Member Data Documentation

8.115.2.1 "@59

```
union { ... }
```

Union of the required information for processing each tensor "format".

8.115.2.2 dimension

```
tensor_dim GstTensorMetaInfo::dimension
```

Definition at line 315 of file tensor_typedef.h.

8.115.2.3 format

```
uint32_t GstTensorMetaInfo::format
```

Definition at line 316 of file tensor_typedef.h.

8.115.2.4 magic

```
uint32_t GstTensorMetaInfo::magic
```

Definition at line 312 of file tensor_typedef.h.

8.115.2.5 media_type

```
uint32_t GstTensorMetaInfo::media_type
```

Definition at line 317 of file tensor_typedef.h.

8.115.2.6 sparse_info

```
GstSparseTensorInfo GstTensorMetaInfo::sparse_info
```

Definition at line 323 of file tensor_typedef.h.

8.115.2.7 type

```
uint32_t GstTensorMetaInfo::type
```

Definition at line 314 of file tensor_typedef.h.

8.115.2.8 version

```
uint32_t GstTensorMetaInfo::version
```

Definition at line 313 of file tensor_typedef.h.

The documentation for this struct was generated from the following file:

- nnstreamer/include/tensor_typedef.h

8.116 GstTensorPad Struct Reference

Internal data structure for pad in demux / split.

`#include <tensor_common.h>`

Public Attributes

- GstPad * pad
- GstClockTime last_ts
- GstFlowReturn last_ret
- guint nth

8.116.1 Detailed Description

Internal data structure for pad in demux / split.

Definition at line 102 of file tensor_common.h.

8.116.2 Member Data Documentation

8.116.2.1 last_ret

`GstFlowReturn GstTensorPad::last_ret`

Definition at line 106 of file tensor_common.h.

8.116.2.2 last_ts

`GstClockTime GstTensorPad::last_ts`

Definition at line 105 of file tensor_common.h.

8.116.2.3 nth

`guint GstTensorPad::nth`

Definition at line 107 of file tensor_common.h.

8.116.2.4 pad

```
GstPad* GstTensorPad::pad
```

Definition at line 104 of file tensor_common.h.

The documentation for this struct was generated from the following file:

- nnstreamer/tensor_common.h

8.117 GstTensorQueryEdgeInfo Struct Reference

Internal data structure for nns-edge info to prepare edge connection.

```
#include <tensor_query_server.h>
```

Public Attributes

- gchar * host
- guint16 port
- gchar * dest_host
- guint16 dest_port
- gchar * topic
- nns_edge_event_cb cb
- void * pdata

8.117.1 Detailed Description

Internal data structure for nns-edge info to prepare edge connection.

Definition at line 28 of file tensor_query_server.h.

8.117.2 Member Data Documentation

8.117.2.1 cb

```
nns_edge_event_cb GstTensorQueryEdgeInfo::cb
```

Definition at line 37 of file tensor_query_server.h.

8.117.2.2 dest_host

```
gchar* GstTensorQueryEdgeInfo::dest_host
```

Definition at line 32 of file tensor_query_server.h.

8.117.2.3 dest_port

```
guint16 GstTensorQueryEdgeInfo::dest_port
```

Definition at line 33 of file tensor_query_server.h.

8.117.2.4 host

```
gchar* GstTensorQueryEdgeInfo::host
```

Definition at line 30 of file tensor_query_server.h.

8.117.2.5 pdata

```
void* GstTensorQueryEdgeInfo::pdata
```

Definition at line 38 of file tensor_query_server.h.

8.117.2.6 port

```
guint16 GstTensorQueryEdgeInfo::port
```

Definition at line 31 of file tensor_query_server.h.

8.117.2.7 topic

```
gchar* GstTensorQueryEdgeInfo::topic
```

Definition at line 34 of file tensor_query_server.h.

The documentation for this struct was generated from the following file:

- nnstreamer/tensor_query/tensor_query_server.h

8.118 GstTensorQueryServer Struct Reference

GstTensorQueryServer internal info data structure.

`#include <tensor_query_server.h>`

Public Attributes

- guint id
- gboolean configured
- GMutex lock
- GCond cond
- nns_edge_h edge_h

8.118.1 Detailed Description

GstTensorQueryServer internal info data structure.

Definition at line 44 of file tensor_query_server.h.

8.118.2 Member Data Documentation

8.118.2.1 cond

`GCond GstTensorQueryServer::cond`

Definition at line 49 of file tensor_query_server.h.

8.118.2.2 configured

`gboolean GstTensorQueryServer::configured`

Definition at line 47 of file tensor_query_server.h.

8.118.2.3 edge_h

`nns_edge_h GstTensorQueryServer::edge_h`

Definition at line 51 of file tensor_query_server.h.

8.118.2.4 id

```
guint GstTensorQueryServer::id
```

Definition at line 46 of file tensor_query_server.h.

8.118.2.5 lock

```
GMutex GstTensorQueryServer::lock
```

Definition at line 48 of file tensor_query_server.h.

The documentation for this struct was generated from the following file:

- nnstreamer/tensor_query/tensor_query_server.h

8.119 GstTensorRepo Struct Reference

GstTensorRepo data structure.

```
#include <gsttensor_repo.h>
```

Public Attributes

- guint num_data
- GMutex repo_lock
- GCond repo_cond
- GHashTable * hash
- gboolean initialized

8.119.1 Detailed Description

GstTensorRepo data structure.

Definition at line 58 of file gsttensor_repo.h.

8.119.2 Member Data Documentation

8.119.2.1 hash

```
GHashTable* GstTensorRepo::hash
```

Definition at line 63 of file gsttensor_repo.h.

8.119.2.2 initialized

```
gboolean GstTensorRepo::initialized
```

Definition at line 64 of file gsttensor_repo.h.

8.119.2.3 num_data

```
guint GstTensorRepo::num_data
```

Definition at line 60 of file gsttensor_repo.h.

8.119.2.4 repo_cond

```
GCond GstTensorRepo::repo_cond
```

Definition at line 62 of file gsttensor_repo.h.

8.119.2.5 repo_lock

```
GMutex GstTensorRepo::repo_lock
```

Definition at line 61 of file gsttensor_repo.h.

The documentation for this struct was generated from the following file:

- nnstreamer/elements/gsttensor_repo.h

8.120 GstTensorRepoData Struct Reference

GstTensorRepo internal data structure.

```
#include <gsttensor_repo.h>
```

Public Attributes

- GstBuffer * buffer
- GstCaps * caps
- GCond cond_push
- GCond cond_pull
- GMutex lock
- gboolean eos
- gboolean src_changed
- guint src_id
- gboolean sink_changed
- guint sink_id
- gboolean pushed

8.120.1 Detailed Description

GstTensorRepo internal data structure.

GstTensorRepo has GSlist of GstTensorRepoData.

Definition at line 40 of file gsttensor_repo.h.

8.120.2 Member Data Documentation

8.120.2.1 buffer

```
GstBuffer* GstTensorRepoData::buffer
```

Definition at line 42 of file gsttensor_repo.h.

8.120.2.2 caps

```
GstCaps* GstTensorRepoData::caps
```

Definition at line 43 of file gsttensor_repo.h.

8.120.2.3 cond_pull

```
GCond GstTensorRepoData::cond_pull
```

Definition at line 45 of file gsttensor_repo.h.

8.120.2.4 cond_push

```
GCond GstTensorRepoData::cond_push
```

Definition at line 44 of file gsttensor_repo.h.

8.120.2.5 eos

```
gboolean GstTensorRepoData::eos
```

Definition at line 47 of file gsttensor_repo.h.

8.120.2.6 lock

```
GMutex GstTensorRepoData::lock
```

Definition at line 46 of file gsttensor_repo.h.

8.120.2.7 pushed

```
gboolean GstTensorRepoData::pushed
```

Definition at line 52 of file gsttensor_repo.h.

8.120.2.8 sink_changed

```
gboolean GstTensorRepoData::sink_changed
```

Definition at line 50 of file gsttensor_repo.h.

8.120.2.9 sink_id

```
guint GstTensorRepoData::sink_id
```

Definition at line 51 of file gsttensor_repo.h.

8.120.2.10 src_changed

```
gboolean GstTensorRepoData::src_changed
```

Definition at line 48 of file gsttensor_repo.h.

8.120.2.11 src_id

```
guint GstTensorRepoData::src_id
```

Definition at line 49 of file gsttensor_repo.h.

The documentation for this struct was generated from the following file:

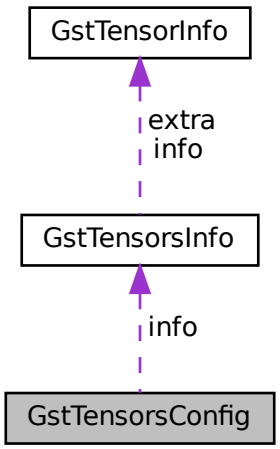
- nnstreamer/elements/gsttensor_repo.h

8.121 GstTensorsConfig Struct Reference

Internal data structure for configured tensors info (for other/tensors).

```
#include <tensor_typedef.h>
```

Collaboration diagram for GstTensorsConfig:



Public Attributes

- GstTensorsInfo info
- int rate_n
- int rate_d

8.121.1 Detailed Description

Internal data structure for configured tensors info (for other/tensors).

Definition at line 284 of file tensor_typedef.h.

8.121.2 Member Data Documentation

8.121.2.1 info

```
GstTensorsInfo GstTensorsConfig::info
```

tensor info

Definition at line 286 of file tensor_typedef.h.

8.121.2.2 rate_d

```
int GstTensorsConfig::rate_d
```

framerate is in fraction, which is numerator/denominator

Definition at line 288 of file tensor_typedef.h.

8.121.2.3 rate_n

```
int GstTensorsConfig::rate_n
```

framerate is in fraction, which is numerator/denominator

Definition at line 287 of file tensor_typedef.h.

The documentation for this struct was generated from the following file:

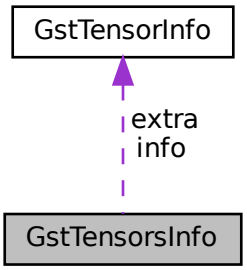
- nnstreamer/include/tensor_typedef.h

8.122 GstTensorsInfo Struct Reference

Internal meta data exchange format for a other/tensors instance.

`#include <tensor_typedef.h>`

Collaboration diagram for GstTensorsInfo:



Public Attributes

- unsigned int num_tensors
- GstTensorInfo info [NNS_TENSOR_MEMORY_MAX]
- GstTensorInfo * extra
- tensor_format format

8.122.1 Detailed Description

Internal meta data exchange format for a other/tensors instance.

Definition at line 273 of file tensor_typedef.h.

8.122.2 Member Data Documentation

8.122.2.1 extra

`GstTensorInfo* GstTensorsInfo::extra`

The list of tensor info for tensors whose index is larger than NNS_TENSOR_MEMORY_MAX

Definition at line 277 of file tensor_typedef.h.

8.122.2.2 format

```
tensor_format GstTensorsInfo::format
```

tensor stream type

Definition at line 278 of file tensor_typedef.h.

8.122.2.3 info

```
GstTensorInfo GstTensorsInfo::info[NNS_TENSOR_MEMORY_MAX]
```

The list of tensor info (max NNS_TENSOR_MEMORY_MAX as static)

Definition at line 276 of file tensor_typedef.h.

8.122.2.4 num_tensors

```
unsigned int GstTensorsInfo::num_tensors
```

The number of tensors

Definition at line 275 of file tensor_typedef.h.

The documentation for this struct was generated from the following file:

- nnstreamer/include/tensor_typedef.h

8.123 parse_accl_args Struct Reference

Accelerator related arguments for parsing.

```
#include <nnstreamer_plugin_api_filter.h>
```

Public Attributes

- const char * in_accl
- const char ** sup_accl
- const char * auto_accl
- const char * def_accl

8.123.1 Detailed Description

Accelerator related arguments for parsing.

Note

if optional accelerators are not set, first entry from supported accelerator is used.

Definition at line 523 of file nnstreamer_plugin_api_filter.h.

8.123.2 Member Data Documentation

8.123.2.1 auto_accl

```
const char* parse_accl_args::auto_accl
```

auto accelerator (optional)

Definition at line 526 of file nnstreamer_plugin_api_filter.h.

8.123.2.2 def_accl

```
const char* parse_accl_args::def_accl
```

default accelerator (optional)

Definition at line 527 of file nnstreamer_plugin_api_filter.h.

8.123.2.3 in_accl

```
const char* parse_accl_args::in_accl
```

user given input

Definition at line 524 of file nnstreamer_plugin_api_filter.h.

8.123.2.4 sup_accl

```
const char** parse_accl_args::sup_accl
```

list of supported accelerator

Definition at line 525 of file nnstreamer_plugin_api_filter.h.

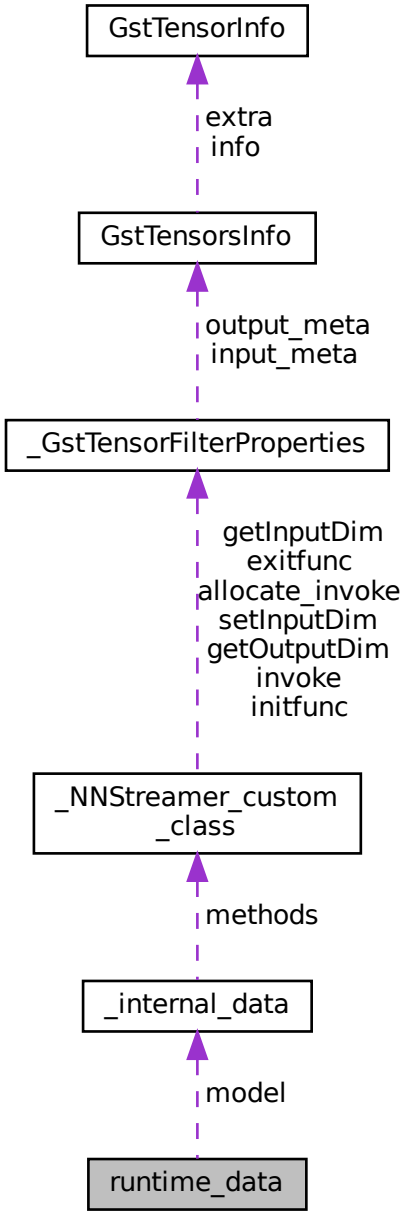
The documentation for this struct was generated from the following file:

- nnstreamer/include/nnstreamer_plugin_api_filter.h

8.124 runtime_data Struct Reference

The easy-filter user's data.

Collaboration diagram for runtime_data:



Public Attributes

- const internal_data * model

8.124.1 Detailed Description

The easy-filter user's data.

Definition at line 53 of file tensor_filter_custom_easy.c.

8.124.2 Member Data Documentation

8.124.2.1 model

```
const internal_data* runtime_data::model
```

Definition at line 55 of file tensor_filter_custom_easy.c.

The documentation for this struct was generated from the following file:

- nnstreamer/tensor_filter/tensor_filter_custom_easy.c

8.125 subplugin_conf Struct Reference

Public Attributes

- gchar * path [CONF_SOURCE_END]
- gchar ** files
- gchar ** names

8.125.1 Detailed Description

Definition at line 82 of file nnstreamer_conf.c.

8.125.2 Member Data Documentation

8.125.2.1 files

```
gchar** subplugin_conf::files
```

Null terminated list of full filepaths

Definition at line 93 of file nnstreamer_conf.c.

8.125.2.2 names

```
gchar** subplugin_conf::names
```

Null terminated list of subplugin names

Definition at line 94 of file nnstreamer_conf.c.

8.125.2.3 path

```
gchar* subplugin_conf::path[CONF_SOURCE_END]
```

directory paths

Definition at line 88 of file nnstreamer_conf.c.

The documentation for this struct was generated from the following file:

- nnstreamer/nnstreamer_conf.c

8.126 subplugin_info_s Struct Reference

```
#include <nnstreamer_conf.h>
```

Public Attributes

- gchar ** names
- gchar ** paths

8.126.1 Detailed Description

Definition at line 75 of file nnstreamer_conf.h.

8.126.2 Member Data Documentation

8.126.2.1 names

```
gchar** subplugin_info_s::names
```

Definition at line 77 of file nnstreamer_conf.h.

8.126.2.2 paths

```
gchar** subplugin_info_s::paths
```

Definition at line 78 of file nnstreamer_conf.h.

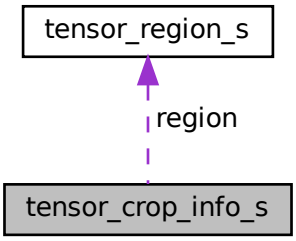
The documentation for this struct was generated from the following file:

- nnstreamer/nnstreamer_conf.h

8.127 tensor_crop_info_s Struct Reference

Internal data structure to describe cropping tensor data.

Collaboration diagram for tensor_crop_info_s:



Public Attributes

- guint num
- tensor_region_s region [NNS_TENSOR_SIZE_LIMIT]

8.127.1 Detailed Description

Internal data structure to describe cropping tensor data.

Todo Add various mode to crop tensor. Now tensor-crop handles NHWC data format only.

Definition at line 62 of file gsttensor_crop.c.

8.127.2 Member Data Documentation

8.127.2.1 num

```
guint tensor_crop_info_s::num
```

Definition at line 64 of file gsttensor_crop.c.

8.127.2.2 region

```
tensor_region_s tensor_crop_info_s::region[NNS_TENSOR_SIZE_LIMIT]
```

Definition at line 65 of file gsttensor_crop.c.

The documentation for this struct was generated from the following file:

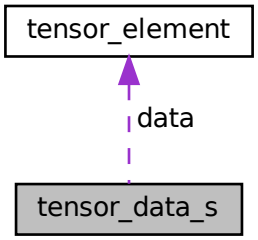
- nnstreamer/elements/gsttensor_crop.c

8.128 tensor_data_s Struct Reference

Structure for tensor data.

```
#include <tensor_data.h>
```

Collaboration diagram for tensor_data_s:



Public Attributes

- tensor_type type
- tensor_element data

8.128.1 Detailed Description

Structure for tensor data.

Definition at line 23 of file tensor_data.h.

8.128.2 Member Data Documentation

8.128.2.1 data

```
tensor_element tensor_data_s::data
```

Definition at line 26 of file tensor_data.h.

8.128.2.2 type

```
tensor_type tensor_data_s::type
```

Definition at line 25 of file tensor_data.h.

The documentation for this struct was generated from the following file:

- nnstreamer/tensor_data.h

8.129 tensor_element Union Reference

To make the code simple with all the types. "C++ Template"-like.

```
#include <tensor_typedef.h>
```

Public Attributes

- int32_t _int32_t
- uint32_t _uint32_t
- int16_t _int16_t
- uint16_t _uint16_t
- int8_t _int8_t
- uint8_t _uint8_t
- double _double
- float _float
- int64_t _int64_t
- uint64_t _uint64_t

8.129.1 Detailed Description

To make the code simple with all the types. "C++ Template"-like.

Definition at line 231 of file tensor_typedef.h.

8.129.2 Member Data Documentation

8.129.2.1 _double

```
double tensor_element::_double
```

Definition at line 238 of file tensor_typedef.h.

8.129.2.2 _float

```
float tensor_element::_float
```

Definition at line 239 of file tensor_typedef.h.

8.129.2.3 _int16_t

```
int16_t tensor_element::_int16_t
```

Definition at line 234 of file tensor_typedef.h.

8.129.2.4 _int32_t

```
int32_t tensor_element::_int32_t
```

Definition at line 232 of file tensor_typedef.h.

8.129.2.5 _int64_t

```
int64_t tensor_element::_int64_t
```

Definition at line 240 of file tensor_typedef.h.

8.129.2.6 _int8_t

```
int8_t tensor_element::_int8_t
```

Definition at line 236 of file tensor_typedef.h.

8.129.2.7 _uint16_t

```
uint16_t tensor_element::_uint16_t
```

Definition at line 235 of file tensor_typedef.h.

8.129.2.8 _uint32_t

```
uint32_t tensor_element::_uint32_t
```

Definition at line 233 of file tensor_typedef.h.

8.129.2.9 _uint64_t

```
uint64_t tensor_element::_uint64_t
```

Definition at line 241 of file tensor_typedef.h.

8.129.2.10 _uint8_t

```
uint8_t tensor_element::_uint8_t
```

Definition at line 237 of file tensor_typedef.h.

The documentation for this union was generated from the following file:

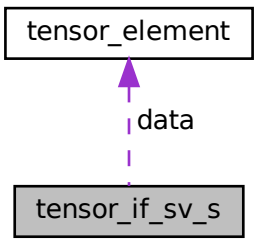
- nnstreamer/include/tensor_typedef.h

8.130 tensor_if_sv_s Struct Reference

Internal data structure for supplied value.

```
#include <gsttensor_if.h>
```

Collaboration diagram for tensor_if_sv_s:



Public Attributes

- guint32 num
- tensor_type type
- tensor_element data [2]

8.130.1 Detailed Description

Internal data structure for supplied value.

Definition at line 105 of file gsttensor_if.h.

8.130.2 Member Data Documentation

8.130.2.1 data

```
tensor_element tensor_if_sv_s::data[2]
```

Definition at line 109 of file gsttensor_if.h.

8.130.2.2 num

```
guint32 tensor_if_sv_s::num
```

Definition at line 107 of file gsttensor_if.h.

8.130.2.3 type

```
tensor_type tensor_if_sv_s::type
```

Definition at line 108 of file gsttensor_if.h.

The documentation for this struct was generated from the following file:

- nnstreamer/elements/gsttensor_if.h

8.131 tensor_region_s Struct Reference

Internal data structure to describe tensor region.

Public Attributes

- guint x
- guint y
- guint w
- guint h

8.131.1 Detailed Description

Internal data structure to describe tensor region.

SECTION:element-tensor_crop

tensor_crop is a GStreamer element to crop the regions of incoming tensor.

tensor_crop has two always sink pads - raw and info. The raw pad accepts tensor (other/tensor) which will be cropped with crop info. The info pad has capability for flexible tensor stream (other/tensors-flexible), that can have a various buffer size for crop info. Incoming buffer on info pad should be an array of crop info. Note that NNStreamer supports maximum NNS_TENSOR_SIZE_LIMIT memory blocks in a buffer. So, when incoming buffer on info pad has more than NNS_TENSOR_SIZE_LIMIT crop-info array, tensor_crop will ignore the data.

The output is always in the format of other/tensors-flexible.

<refsect2> <title>Example launch line</title> |[gst-launch-1.0 tensor_crop name=crop ! (cropped tensors) ... \ videotestsrc ! videoconvert ! video/x-raw,format=RGB ! tensor_converter ! tee name=t \ t. ! queue ! crop.raw \ t. ! queue ! (process raw video tensor and push buffer which includes crop info) ! crop.info]| </refsect2>

Definition at line 50 of file gsttensor_crop.c.

8.131.2 Member Data Documentation

8.131.2.1 h

```
guint tensor_region_s::h
```

Definition at line 55 of file gsttensor_crop.c.

8.131.2.2 w

```
guint tensor_region_s::w
```

Definition at line 54 of file gsttensor_crop.c.

8.131.2.3 x

```
guint tensor_region_s::x
```

Definition at line 52 of file gsttensor_crop.c.

8.131.2.4 y

```
guint tensor_region_s::y
```

Definition at line 53 of file gsttensor_crop.c.

The documentation for this struct was generated from the following file:

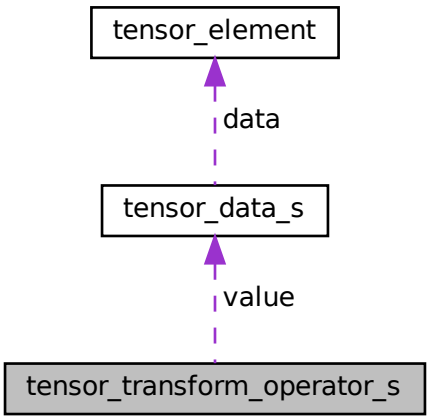
- nnstreamer/elements/gsttensor_crop.c

8.132 tensor_transform_operator_s Struct Reference

Internal data structure for operator of arithmetic mode.

```
#include <gsttensor_transform.h>
```

Collaboration diagram for tensor_transform_operator_s:



Public Attributes

- tensor_transform_operator op
- int applying_ch
- tensor_data_s value

8.132.1 Detailed Description

Internal data structure for operator of arithmetic mode.

Definition at line 116 of file gsttensor_transform.h.

8.132.2 Member Data Documentation

8.132.2.1 applying_ch

```
int tensor_transform_operator_s::applying_ch
```

Definition at line 119 of file gsttensor_transform.h.

8.132.2.2 op

```
tensor_transform_operator tensor_transform_operator_s::op
```

Definition at line 118 of file gsttensor_transform.h.

8.132.2.3 value

```
tensor_data_s tensor_transform_operator_s::value
```

Definition at line 120 of file gsttensor_transform.h.

The documentation for this struct was generated from the following file:

- nnstreamer/elements/gsttensor_transform.h

8.133 vstr_helper Struct Reference

Data structure for _g_list_foreach_vstr_helper.

Public Attributes

- gchar ** vstr
- guint cursor
- guint size

8.133.1 Detailed Description

Data structure for _g_list_foreach_vstr_helper.

Definition at line 255 of file nnstreamer_conf.c.

8.133.2 Member Data Documentation

8.133.2.1 cursor

```
guint vstr_helper::cursor
```

The first "empty" element in vstr

Definition at line 258 of file nnstreamer_conf.c.

8.133.2.2 size

```
guint vstr_helper::size
```

The number of "g_char *" in vstr, excluding the terminator

Definition at line 259 of file nnstreamer_conf.c.

8.133.2.3 vstr

```
gchar** vstr_helper::vstr
```

The vstr data (string array)

Definition at line 257 of file nnstreamer_conf.c.

The documentation for this struct was generated from the following file:

- nnstreamer/nnstreamer_conf.c

Chapter 9

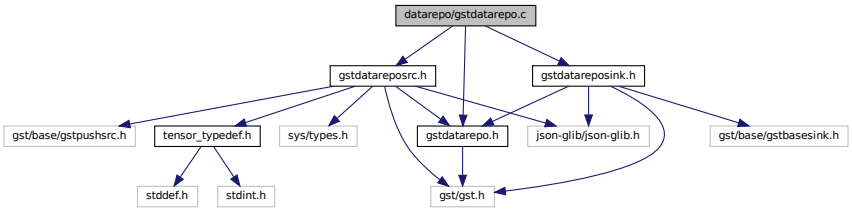
File Documentation

9.1 datarepo/gstdatarepo.c File Reference

Register datarepo plugins.

```
#include "gstdatarepo.h"
#include "gstdatareposrc.h"
#include "gstdatareposink.h"
```

Include dependency graph for gstdatarepo.c:



Macros

- #define PACKAGE "NNStreamer MLOps Data Repository Plugins"

Functions

- GstDataRepoDataType gst_data_repo_get_data_type_from_caps (const GstCaps ∗caps)

  Get data type from caps.
- static gboolean plugin_init (GstPlugin ∗plugin)

  The entry point of the Gstreamer datarepo plugin.

9.1.1 Detailed Description

Register datarepo plugins.

Date

31 January 2023

See also

https://github.com/nnstreamer/nnstreamer

Author

Hyunil Park hyunil46.park@samsung.com

Bug No known bugs except for NYI items

9.1.2 Macro Definition Documentation

9.1.2.1 PACKAGE

```
#define PACKAGE "NNStreamer MLOps Data Repository Plugins"
```

Definition at line 69 of file gstdatarepo.c.

9.1.3 Function Documentation

9.1.3.1 gst_data_repo_get_data_type_from_caps()

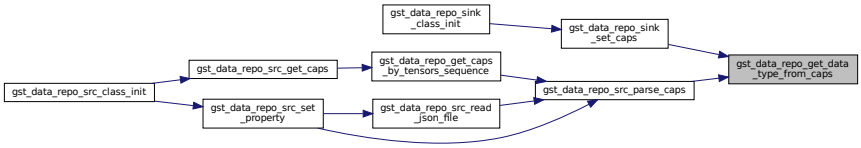
```
GstDataRepoDataType gst_data_repo_get_data_type_from_caps (
            const GstCaps * caps )
```

Get data type from caps.

Definition at line 21 of file gstdatarepo.c.

Here is the call graph for this function:

Here is the caller graph for this function:



9.1.3.2 plugin_init()

```
static gboolean plugin_init (
            GstPlugin * plugin )  [static]
```

The entry point of the Gstreamer datarepo plugin.

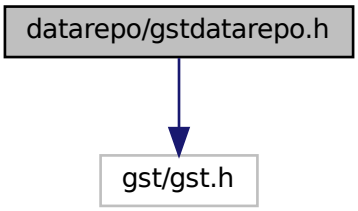
Definition at line 55 of file gstdatarepo.c.

9.2 datarepo/gstdatarepo.h File Reference

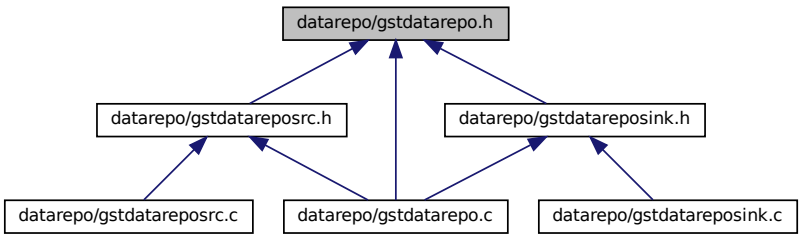
GStreamer plugin to read file in MLOps Data repository into buffers.

```
#include <gst/gst.h>
```

Include dependency graph for gstdatarepo.h:



This graph shows which files directly or indirectly include this file:



Enumerations

- enum GstDataRepoDataType {
  GST_DATA_REPO_DATA_UNKNOWN = 0, GST_DATA_REPO_DATA_VIDEO, GST_DATA_REPO_DATA_AUDIO,
  GST_DATA_REPO_DATA_TEXT,
  GST_DATA_REPO_DATA_OCTET, GST_DATA_REPO_DATA_TENSOR, GST_DATA_REPO_DATA_IMAGE,
  GST_DATA_REPO_DATA_MAX }

  Data type of incoming buffer.

Functions

- GstDataRepoDataType gst_data_repo_get_data_type_from_caps (const GstCaps *caps)

  Get data type from caps.

9.2.1 Detailed Description

GStreamer plugin to read file in MLOps Data repository into buffers.

Date

27 June 2023

See also

https://github.com/nnstreamer/nnstreamer

Author

Hyunil Park hyunil46.park@samsung.com

Bug No known bugs except for NYI items

9.2.2 Enumeration Type Documentation

9.2.2.1 GstDataRepoDataType

```
enum GstDataRepoDataType
```

Data type of incoming buffer.

Enumerator

GST_DATA_REPO_DATA_UNKNOWN	
GST_DATA_REPO_DATA_VIDEO	
GST_DATA_REPO_DATA_AUDIO	
GST_DATA_REPO_DATA_TEXT	
GST_DATA_REPO_DATA_OCTET	
GST_DATA_REPO_DATA_TENSOR	
GST_DATA_REPO_DATA_IMAGE	
GST_DATA_REPO_DATA_MAX	

Definition at line 23 of file gstdatarepo.h.

9.2.3 Function Documentation

9.2.3.1 gst_data_repo_get_data_type_from_caps()

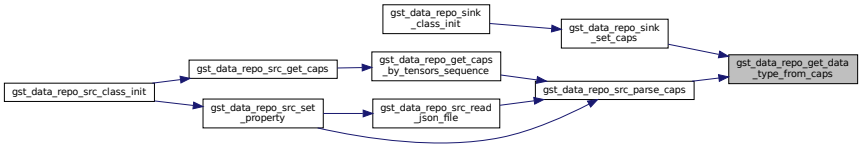
```
GstDataRepoDataType gst_data_repo_get_data_type_from_caps (
            const GstCaps * caps )
```

Get data type from caps.

Definition at line 21 of file gstdatarepo.c.

Here is the call graph for this function:

Here is the caller graph for this function:

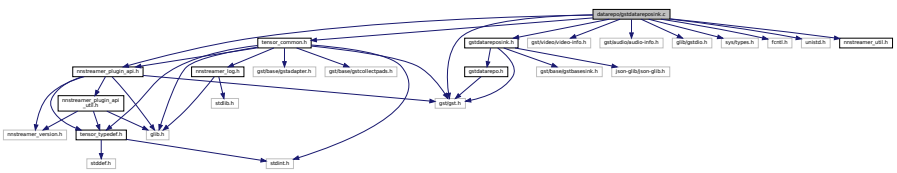


9.3 datarepo/gstdatareposink.c File Reference

GStreamer plugin that writes data from buffers to files in in MLOps Data repository.

```
#include <gst/gst.h>
#include <gst/video/video-info.h>
#include <gst/audio/audio-info.h>
#include <glib/gstdio.h>
#include <sys/types.h>
#include <fcntl.h>
#include <unistd.h>
#include <nnstreamer_plugin_api.h>
#include <tensor_common.h>
#include <nnstreamer_util.h>
#include "gstdatareposink.h"
```

Include dependency graph for gstdatareposink.c:



Macros

- #define TENSOR_CAPS GST_TENSORS_CAP_MAKE ("{ static, flexible, sparse }")

  Tensors caps.
- #define SUPPORTED_VIDEO_FORMAT "{RGB, BGR, RGBx, BGRx, xRGB, xBGR, RGBA, BGRA, ARGB, ABGR, GRAY8}"

  Video caps.
- #define VIDEO_CAPS
- #define SUPPORTED_AUDIO_FORMAT "{S8, U8, S16LE, S16BE, U16LE, U16BE, S32LE, S32BE, U32LE, U32BE, F32LE, F32BE, F64LE, F64BE}"

  Audio caps.
- #define AUDIO_CAPS
- #define TEXT_CAPS "text/x-raw, format = (string) utf8"

  Text caps.
- #define OCTET_CAPS "application/octet-stream"

  Octet caps.
- #define IMAGE_CAPS

  Image caps.
- #define GST_CAT_DEFAULT gst_data_repo_sink_debug
- #define _do_init GST_DEBUG_CATEGORY_INIT (gst_data_repo_sink_debug, "datareposink", 0, "datareposink element");
- #define gst_data_repo_sink_parent_class parent_class

Enumerations

- enum { PROP_0, PROP_LOCATION, PROP_JSON }

  datareposink properties.

Functions

- GST_DEBUG_CATEGORY_STATIC (gst_data_repo_sink_debug)
- G_DEFINE_TYPE_WITH_CODE (GstDataRepoSink, gst_data_repo_sink, GST_TYPE_BASE_SINK, _do_init)
- static void gst_data_repo_sink_set_property (GObject *object, guint prop_id, const GValue *value, GParamSpec *pspec)

  Setter for datareposink properties.
- static void gst_data_repo_sink_get_property (GObject *object, guint prop_id, GValue *value, GParamSpec *pspec)

  Getter datareposink properties.
- static void gst_data_repo_sink_finalize (GObject *object)

  finalize datareposink.
- static gboolean gst_data_repo_sink_stop (GstBaseSink *basesink)

  Stop datareposink.
- static GstStateChangeReturn gst_data_repo_sink_change_state (GstElement *element, GstStateChange transition)

  Change state of datareposink.
- static GstFlowReturn gst_data_repo_sink_render (GstBaseSink *bsink, GstBuffer *buffer)

  Called when a buffer should be presented or output.
- static GstCaps * gst_data_repo_sink_get_caps (GstBaseSink *bsink, GstCaps *filter)

  Get caps of datareposink.
- static gboolean gst_data_repo_sink_set_caps (GstBaseSink *bsink, GstCaps *caps)

Set caps of datareposink.

- static gboolean gst_data_repo_sink_query (GstBaseSink *bsink, GstQuery *query)

  Perform a GstQuery on datareposink.
- static void gst_data_repo_sink_class_init (GstDataRepoSinkClass *klass)

  Initialize datareposink class.
- static void gst_data_repo_sink_init (GstDataRepoSink *sink)

  Initialize datareposink.
- static GstFlowReturn gst_data_repo_sink_write_others (GstDataRepoSink *sink, GstBuffer *buffer)

  Function to write others media type (tensors(fixed), video, audio, octet and text)
- static GstFlowReturn gst_data_repo_sink_write_flexible_or_sparse_tensors (GstDataRepoSink *sink, GstBuffer *buffer)

  Function to write flexible tensors or sparse tensors.
- static gchar * gst_data_repo_sink_get_image_filename (GstDataRepoSink *sink)

  Get image filename.
- static GstFlowReturn gst_data_repo_sink_write_multi_images (GstDataRepoSink *sink, GstBuffer *buffer)

  Function to read multi image files.
- static void gst_data_repo_sink_set_is_static_tensors (GstDataRepoSink *sink)

  Set whether the given pad caps are static or not.
- static gboolean gst_data_repo_sink_open_file (GstDataRepoSink *sink)

  Function to open file.
- static gboolean __write_json (JsonObject *object, const gchar *filename)

  Write json to file.
- static gboolean gst_data_repo_sink_write_json_meta_file (GstDataRepoSink *sink)

  write the meta information to a JSON file

Variables

- static GstStaticPadTemplate sinktemplate

9.3.1 Detailed Description

GStreamer plugin that writes data from buffers to files in in MLOps Data repository.

Date

30 March 2023

See also

https://github.com/nnstreamer/nnstreamer

Author

Hyunil Park hyunil46.park@samsung.com

Bug No known bugs except for NYI items

//! Example launch line |[gst-launch-1.0 videotestsrc ! datareposink location=filename json=video.json gst-launch-1.0 videotestsrc ! pngenc ! datareposink location=image_%02d.png json=video.json gst-launch-1.0 audiotestsrc samplesperbuffer=44100 ! audio/x-raw, format=S16LE, layout=interleaved, rate=44100, channels=1 ! \ datareposink location=filename json=audio.json gst-launch-1.0 datareposrc location=file.dat json=file.json tensors-sequence=2,3 start-sample-index=0 stop-sample-index=199 epochs=1 ! \ other/tensors, format=static, num_↩ tensors=2, framerate=0/1, dimensions=1:1:784:1.1:1:10:1, types=float32.float32 ! \ datareposink location=hyunil.↩ dat json=file.json]|

Date

30 March 2023

See also

https://github.com/nnstreamer/nnstreamer

Author

Hyunil Park hyunil46.park@samsung.com

Bug No known bugs except for NYI items

9.3.2 Macro Definition Documentation

9.3.2.1 _do_init

```
#define _do_init GST_DEBUG_CATEGORY_INIT (gst_data_repo_sink_debug, "datareposink", 0, "datareposink element");
```

Definition at line 92 of file gstdatareposink.c.

9.3.2.2 AUDIO_CAPS

```
#define AUDIO_CAPS
```

Value:

```
  GST_AUDIO_CAPS_MAKE (SUPPORTED_AUDIO_FORMAT) "," \
  "layout = (string) interleaved"
```

Definition at line 55 of file gstdatareposink.c.

9.3.2.3 GST_CAT_DEFAULT

```
#define GST_CAT_DEFAULT gst_data_repo_sink_debug
```

Definition at line 91 of file gstdatareposink.c.

9.3.2.4 gst_data_repo_sink_parent_class

```
#define gst_data_repo_sink_parent_class parent_class
```

Definition at line 94 of file gstdatareposink.c.

9.3.2.5 IMAGE_CAPS

```
#define IMAGE_CAPS
```

Value:

```
  "image/png, width = (int) [ 16, 1000000 ], height = (int) [ 16, 1000000 ], framerate = (fraction) [ 0/1,
       MAX];" \
  "image/jpeg, width = (int) [ 16, 65535 ], height = (int) [ 16, 65535 ], framerate = (fraction) [ 0/1,
       MAX], sof-marker = (int) { 0, 1, 2, 4, 9 };" \
  "image/tiff, endianness = (int) { BIG_ENDIAN, LITTLE_ENDIAN };" \
  "image/gif;" \
  "image/bmp"
```

Image caps.

Definition at line 68 of file gstdatareposink.c.

9.3.2.6 OCTET_CAPS

```
#define OCTET_CAPS "application/octet-stream"
```

Octet caps.

Definition at line 64 of file gstdatareposink.c.

9.3.2.7 SUPPORTED_AUDIO_FORMAT

```
#define SUPPORTED_AUDIO_FORMAT "{S8, U8, S16LE, S16BE, U16LE, U16BE, S32LE, S32BE, U32LE,
U32BE, F32LE, F32BE, F64LE, F64BE}"
```

Audio caps.

Definition at line 53 of file gstdatareposink.c.

9.3.2.8 SUPPORTED_VIDEO_FORMAT

```
#define SUPPORTED_VIDEO_FORMAT "{RGB, BGR, RGBx, BGRx, xRGB, xBGR, RGBA, BGRA, ARGB, ABGR, G↩
RAY8}"
```

Video caps.

Definition at line 46 of file gstdatareposink.c.

9.3.2.9 TENSOR_CAPS

```
#define TENSOR_CAPS GST_TENSORS_CAP_MAKE ("{ static, flexible, sparse }")
```

Tensors caps.

Definition at line 42 of file gstdatareposink.c.

9.3.2.10 TEXT_CAPS

```
#define TEXT_CAPS "text/x-raw, format = (string) utf8"
```

Text caps.

Definition at line 60 of file gstdatareposink.c.

9.3.2.11 VIDEO_CAPS

```
#define VIDEO_CAPS
```

Value:

```
  GST_VIDEO_CAPS_MAKE (SUPPORTED_VIDEO_FORMAT) "," \
  "interlace-mode = (string) progressive"
```

Definition at line 48 of file gstdatareposink.c.

9.3.3 Enumeration Type Documentation

9.3.3.1 anonymous enum

```
anonymous enum
```

datareposink properties.

Enumerator

PROP_0	
PROP_LOCATION	
PROP_JSON	

Definition at line 83 of file gstdatareposink.c.

9.3.4 Function Documentation

9.3.4.1 __write_json()

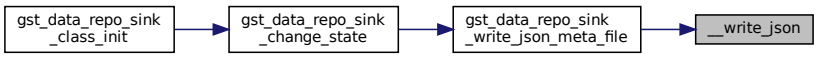
```
static gboolean __write_json (
            JsonObject * object,
            const gchar * filename )  [static]
```

Write json to file.

Definition at line 668 of file gstdatareposink.c.

Here is the call graph for this function:

Here is the caller graph for this function:



9.3.4.2 G_DEFINE_TYPE_WITH_CODE()

```
G_DEFINE_TYPE_WITH_CODE (
            GstDataRepoSink ,
            gst_data_repo_sink ,
            GST_TYPE_BASE_SINK ,
            _do_init  )
```

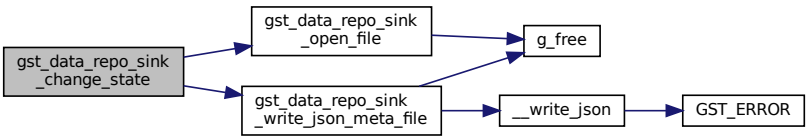
9.3.4.3 gst_data_repo_sink_change_state()

```
static GstStateChangeReturn gst_data_repo_sink_change_state (
            GstElement * element,
            GstStateChange transition )  [static]
```

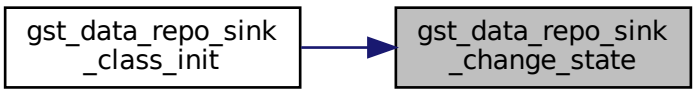
Change state of datareposink.

Definition at line 750 of file gstdatareposink.c.

Here is the call graph for this function:



Here is the caller graph for this function:



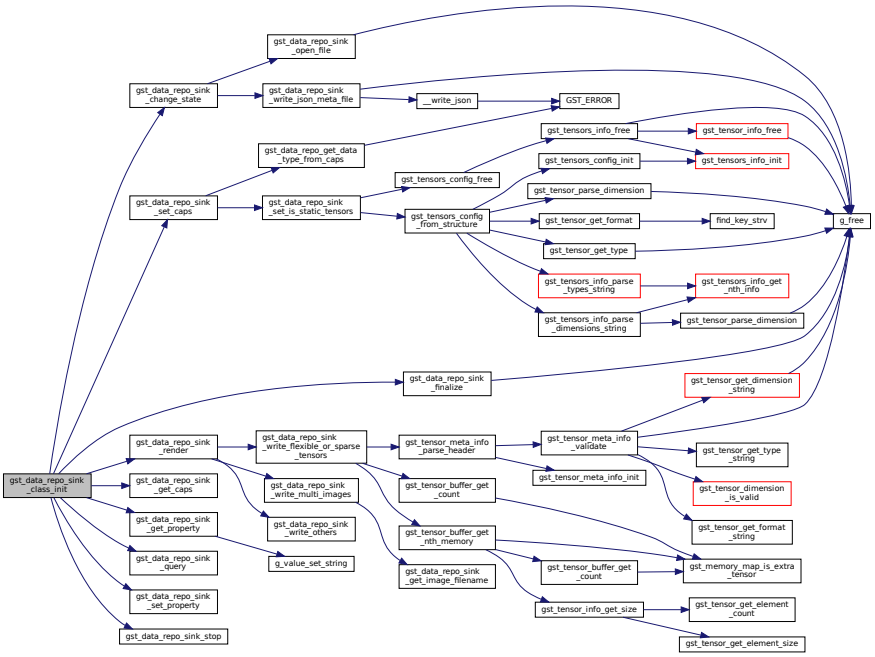
9.3.4.4 gst_data_repo_sink_class_init()

```
static void gst_data_repo_sink_class_init (
            GstDataRepoSinkClass * klass )  [static]
```

Initialize datareposink class.

Definition at line 118 of file gstdatareposink.c.

Here is the call graph for this function:



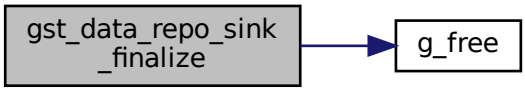
9.3.4.5 gst_data_repo_sink_finalize()

```
static void gst_data_repo_sink_finalize (
            GObject * object )  [static]
```

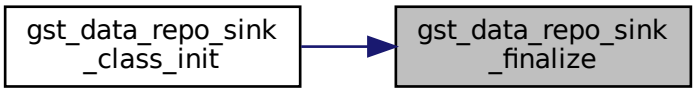
finalize datareposink.

Definition at line 195 of file gstdatareposink.c.

Here is the call graph for this function:



Here is the caller graph for this function:



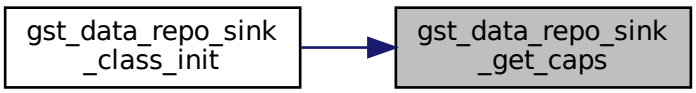
9.3.4.6 gst_data_repo_sink_get_caps()

```
static GstCaps * gst_data_repo_sink_get_caps (
            GstBaseSink * bsink,
            GstCaps * filter )  [static]
```

Get caps of datareposink.

Definition at line 494 of file gstdatareposink.c.

Here is the caller graph for this function:



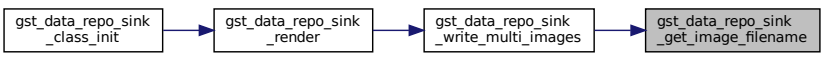
9.3.4.7 gst_data_repo_sink_get_image_filename()

```
static gchar* gst_data_repo_sink_get_image_filename (
            GstDataRepoSink * sink )  [static]
```

Get image filename.

Definition at line 400 of file gstdatareposink.c.

Here is the caller graph for this function:



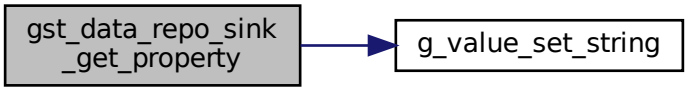
9.3.4.8 gst_data_repo_sink_get_property()

```
static void gst_data_repo_sink_get_property (
             GObject * object,
             guint prop_id,
             GValue * value,
             GParamSpec * pspec )  [static]
```

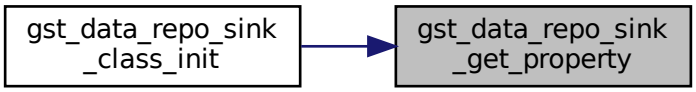
Getter datareposink properties.

Definition at line 251 of file gstdatareposink.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.3.4.9 gst_data_repo_sink_init()

```
static void gst_data_repo_sink_init (
             GstDataRepoSink * sink )  [static]
```

Initialize datareposink.

Definition at line 174 of file gstdatareposink.c.

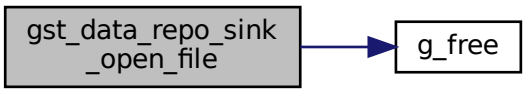
9.3.4.10 gst_data_repo_sink_open_file()

```
static gboolean gst_data_repo_sink_open_file (
            GstDataRepoSink * sink )  [static]
```

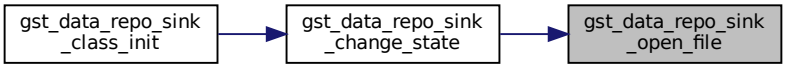
Function to open file.

Definition at line 590 of file gstdatareposink.c.

Here is the call graph for this function:



Here is the caller graph for this function:



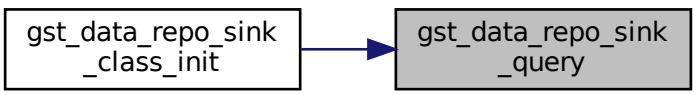
9.3.4.11 gst_data_repo_sink_query()

```
static gboolean gst_data_repo_sink_query (
            GstBaseSink * sink,
            GstQuery * query )  [static]
```

Perform a GstQuery on datareposink.

Definition at line 564 of file gstdatareposink.c.

Here is the caller graph for this function:



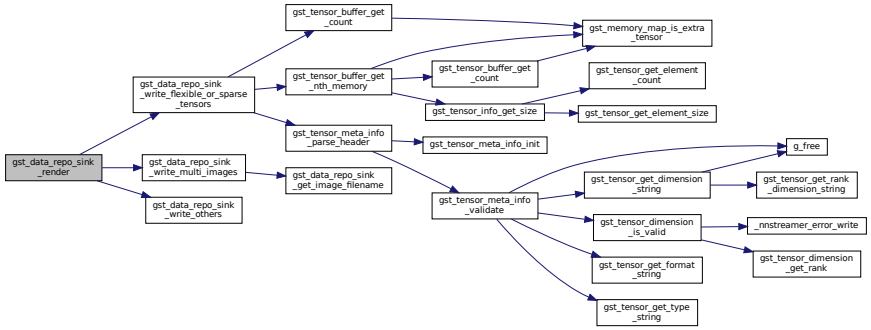
9.3.4.12 gst_data_repo_sink_render()

```
static GstFlowReturn gst_data_repo_sink_render (
            GstBaseSink * bsink,
            GstBuffer * buffer )  [static]
```

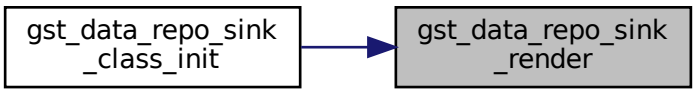
Called when a buffer should be presented or output.

Definition at line 467 of file gstdatareposink.c.

Here is the call graph for this function:



Here is the caller graph for this function:



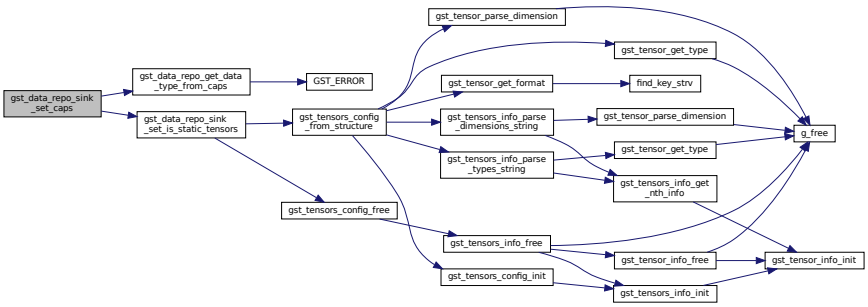
9.3.4.13 gst_data_repo_sink_set_caps()

```
static gboolean gst_data_repo_sink_set_caps (
            GstBaseSink * bsink,
            GstCaps * caps )  [static]
```

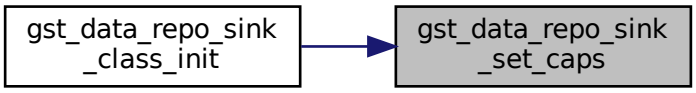
Set caps of datareposink.

Definition at line 539 of file gstdatareposink.c.

Here is the call graph for this function:



Here is the caller graph for this function:



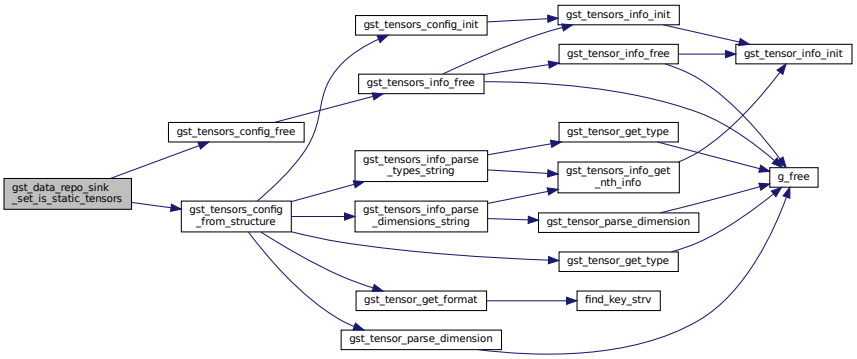
9.3.4.14 gst_data_repo_sink_set_is_static_tensors()

```
static void gst_data_repo_sink_set_is_static_tensors (
            GstDataRepoSink * sink )  [static]
```

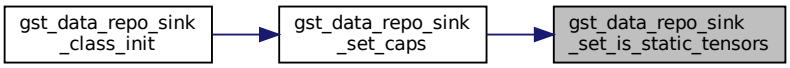
Set whether the given pad caps are static or not.

Definition at line 520 of file gstdatareposink.c.

Here is the call graph for this function:



Here is the caller graph for this function:



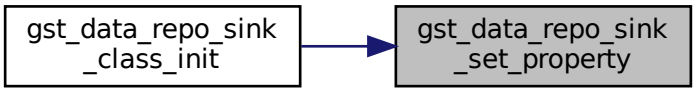
9.3.4.15 gst_data_repo_sink_set_property()

```
static void gst_data_repo_sink_set_property (
            GObject * object,
            guint prop_id,
            const GValue * value,
            GParamSpec * pspec )  [static]
```

Setter for datareposink properties.

Definition at line 227 of file gstdatareposink.c.

Here is the caller graph for this function:



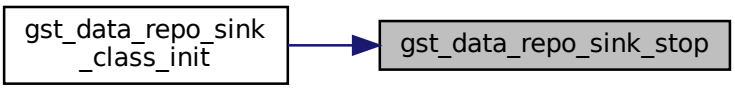
9.3.4.16 gst_data_repo_sink_stop()

```
static gboolean gst_data_repo_sink_stop (
            GstBaseSink * basesink )  [static]
```

Stop datareposink.

Definition at line 652 of file gstdatareposink.c.

Here is the caller graph for this function:



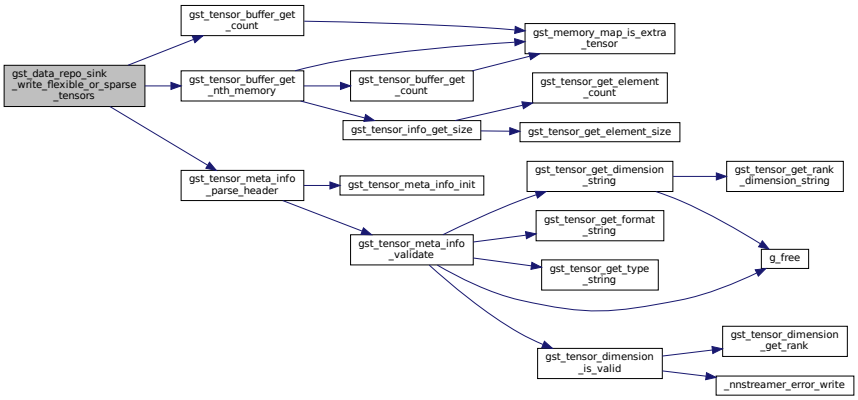
9.3.4.17 gst_data_repo_sink_write_flexible_or_sparse_tensors()

```
static GstFlowReturn gst_data_repo_sink_write_flexible_or_sparse_tensors (
            GstDataRepoSink * sink,
            GstBuffer * buffer )  [static]
```

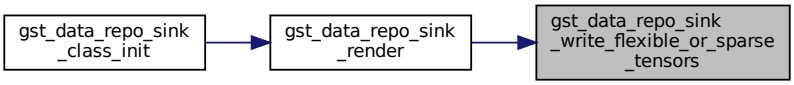
Function to write flexible tensors or sparse tensors.

Definition at line 317 of file gstdatareposink.c.

Here is the call graph for this function:



Here is the caller graph for this function:



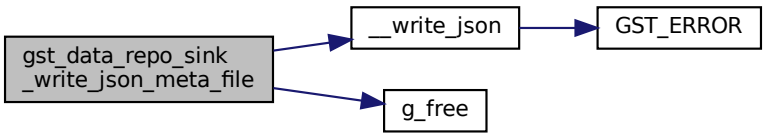
9.3.4.18 gst_data_repo_sink_write_json_meta_file()

```
static gboolean gst_data_repo_sink_write_json_meta_file (
            GstDataRepoSink * sink )  [static]
```

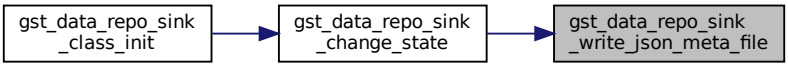
write the meta information to a JSON file

Definition at line 696 of file gstdatareposink.c.

Here is the call graph for this function:



Here is the caller graph for this function:



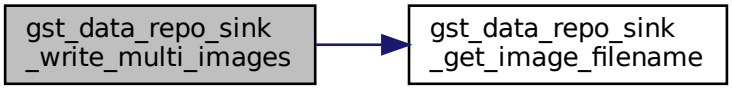
9.3.4.19 gst_data_repo_sink_write_multi_images()

```
static GstFlowReturn gst_data_repo_sink_write_multi_images (
            GstDataRepoSink * sink,
            GstBuffer * buffer )  [static]
```

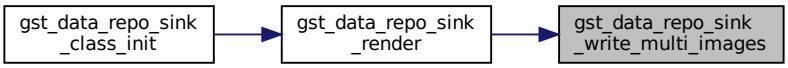
Function to read multi image files.

Definition at line 424 of file gstdatareposink.c.

Here is the call graph for this function:



Here is the caller graph for this function:



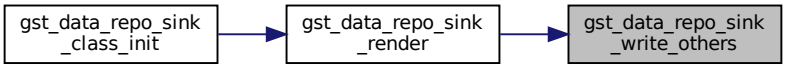
9.3.4.20 gst_data_repo_sink_write_others()

```
static GstFlowReturn gst_data_repo_sink_write_others (
            GstDataRepoSink * sink,
            GstBuffer * buffer )  [static]
```

Function to write others media type (tensors(fixed), video, audio, octet and text)

Definition at line 275 of file gstdatareposink.c.

Here is the caller graph for this function:



9.3.4.21 GST_DEBUG_CATEGORY_STATIC()

```
GST_DEBUG_CATEGORY_STATIC (
            gst_data_repo_sink_debug  )
```

9.3.5 Variable Documentation

9.3.5.1 sinktemplate

```
GstStaticPadTemplate sinktemplate  [static]
```

Initial value:

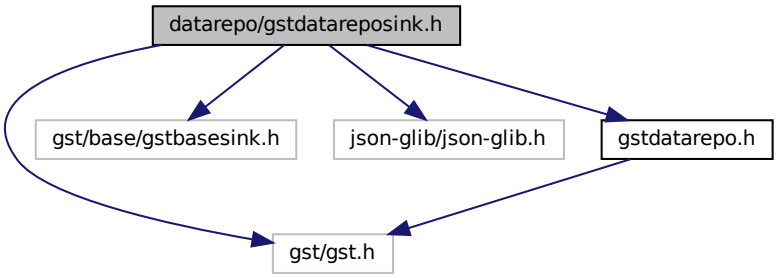
```
=
    GST_STATIC_PAD_TEMPLATE ("sink", GST_PAD_SINK, GST_PAD_ALWAYS,
    GST_STATIC_CAPS (TENSOR_CAPS ";" VIDEO_CAPS ";" AUDIO_CAPS ";" IMAGE_CAPS
        ";" TEXT_CAPS ";" OCTET_CAPS))
```

Definition at line 75 of file gstdatareposink.c.

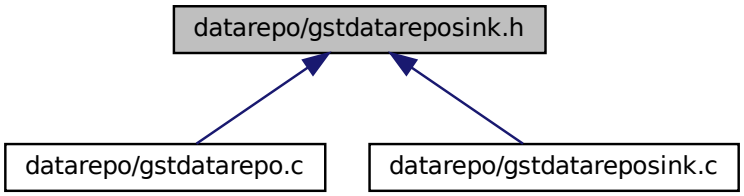
9.4 datarepo/gstdatareposink.h File Reference

```
#include <gst/gst.h>
#include <gst/base/gstbasesink.h>
#include <json-glib/json-glib.h>
#include "gstdatarepo.h"
```

Include dependency graph for gstdatareposink.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct _GstDataRepoSink

  GstDataRepoSink data structure.
- struct _GstDataRepoSinkClass

  GstDataRepoSinkClass data structure.

Macros

- #define GST_TYPE_DATA_REPO_SINK (gst_data_repo_sink_get_type())
- #define GST_DATA_REPO_SINK(obj) (G_TYPE_CHECK_INSTANCE_CAST((obj),GST_TYPE_DATA_REPO_SINK,GstData
- #define GST_DATA_REPO_SINK_CLASS(klass) (G_TYPE_CHECK_CLASS_CAST((klass),GST_TYPE_DATA_REPO_SINK
- #define GST_IS_DATA_REPO_SINK(obj) (G_TYPE_CHECK_INSTANCE_TYPE((obj),GST_TYPE_DATA_REPO_SINK))
- #define GST_IS_DATA_REPO_SINK_CLASS(klass) (G_TYPE_CHECK_CLASS_TYPE((klass),GST_TYPE_DATA_REPO_SI
- #define GST_DATA_REPO_SINK_CAST(obj) ((GstDataRepoSink *)obj)

Typedefs

- typedef struct _GstDataRepoSink GstDataRepoSink
- typedef struct _GstDataRepoSinkClass GstDataRepoSinkClass

Functions

- GType gst_data_repo_sink_get_type (void)

9.4.1 Macro Definition Documentation

9.4.1.1 GST_DATA_REPO_SINK

```
#define GST_DATA_REPO_SINK(
              obj ) (G_TYPE_CHECK_INSTANCE_CAST((obj),GST_TYPE_DATA_REPO_SINK,GstDataRepoSink))
```

Definition at line 24 of file gstdatareposink.h.

9.4.1.2 GST_DATA_REPO_SINK_CAST

```
#define GST_DATA_REPO_SINK_CAST(
              obj ) ((GstDataRepoSink *)obj)
```

Definition at line 32 of file gstdatareposink.h.

9.4.1.3 GST_DATA_REPO_SINK_CLASS

```
#define GST_DATA_REPO_SINK_CLASS(
              klass ) (G_TYPE_CHECK_CLASS_CAST((klass),GST_TYPE_DATA_REPO_SINK,GstDataRepoSinkClass))
```

Definition at line 26 of file gstdatareposink.h.

9.4.1.4 GST_IS_DATA_REPO_SINK

```
#define GST_IS_DATA_REPO_SINK(
              obj ) (G_TYPE_CHECK_INSTANCE_TYPE((obj),GST_TYPE_DATA_REPO_SINK))
```

Definition at line 28 of file gstdatareposink.h.

9.4.1.5 GST_IS_DATA_REPO_SINK_CLASS

```
#define GST_IS_DATA_REPO_SINK_CLASS(
             klass ) (G_TYPE_CHECK_CLASS_TYPE((klass),GST_TYPE_DATA_REPO_SINK))
```

Definition at line 30 of file gstdatareposink.h.

9.4.1.6 GST_TYPE_DATA_REPO_SINK

```
#define GST_TYPE_DATA_REPO_SINK (gst_data_repo_sink_get_type())
```

Definition at line 22 of file gstdatareposink.h.

9.4.2 Typedef Documentation

9.4.2.1 GstDataRepoSink

```
typedef struct _GstDataRepoSink GstDataRepoSink
```

Definition at line 34 of file gstdatareposink.h.

9.4.2.2 GstDataRepoSinkClass

```
typedef struct _GstDataRepoSinkClass GstDataRepoSinkClass
```

Definition at line 35 of file gstdatareposink.h.

9.4.3 Function Documentation

9.4.3.1 gst_data_repo_sink_get_type()

```
GType gst_data_repo_sink_get_type (
            void  )
```

9.5 datarepo/gstdatareposrc.c File Reference

GStreamer plugin to read file in MLOps Data repository into buffers.

```
#include <gst/gst.h>
#include <gst/video/video-info.h>
#include <gst/audio/audio-info.h>
#include <glib/gstdio.h>
#include <nnstreamer_plugin_api.h>
#include <nnstreamer_util.h>
#include <stdio.h>
#include <fcntl.h>
#include <unistd.h>
#include <errno.h>
#include <string.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <inttypes.h>
#include "gstdatareposrc.h"
```

Include dependency graph for gstdatareposrc.c:

Macros

- #define struct_stat struct stat
- #define S_ISREG(mode) ((mode)&_S_IFREG)
- #define S_ISDIR(mode) ((mode)&_S_IFDIR)
- #define S_ISSOCK(x) (0)
- #define O_BINARY (0)
- #define GST_CAT_DEFAULT gst_data_repo_src_debug
- #define DEFAULT_INDEX 0
- #define DEFAULT_EPOCHS 1
- #define DEFAULT_IS_SHUFFLE TRUE
- #define _do_init GST_DEBUG_CATEGORY_INIT (gst_data_repo_src_debug, "datareposrc", 0, "datareposrc element");
- #define gst_data_repo_src_parent_class parent_class

Enumerations

- enum {
  PROP_0, PROP_LOCATION, PROP_JSON, PROP_START_SAMPLE_INDEX,
  PROP_STOP_SAMPLE_INDEX, PROP_EPOCHS, PROP_IS_SHUFFLE, PROP_TENSORS_SEQUENCE,
  PROP_CAPS }

Functions

- GST_DEBUG_CATEGORY_STATIC (gst_data_repo_src_debug)
- static void gst_data_repo_src_finalize (GObject *object)

  Function to finalize instance.
- static GstStateChangeReturn gst_data_repo_src_change_state (GstElement *element, GstStateChange transition)

  Change state of datareposrc.
- static void gst_data_repo_src_set_property (GObject *object, guint prop_id, const GValue *value, GParamSpec *pspec)

  Setter for datareposrc properties.
- static void gst_data_repo_src_get_property (GObject *object, guint prop_id, GValue *value, GParamSpec *pspec)

  Getter datareposrc properties.
- static gboolean gst_data_repo_src_stop (GstBaseSrc *basesrc)

  Stop datareposrc, unmap and close the file.
- static GstCaps * gst_data_repo_src_get_caps (GstBaseSrc *basesrc, GstCaps *filter)

  Get caps for caps negotiation.
- static gboolean gst_data_repo_src_set_caps (GstBaseSrc *basesrc, GstCaps *caps)

  caps after caps negotiation
- static GstFlowReturn gst_data_repo_src_create (GstPushSrc *pushsrc, GstBuffer **buffer)

  Function to create a buffer.
- G_DEFINE_TYPE_WITH_CODE (GstDataRepoSrc, gst_data_repo_src, GST_TYPE_PUSH_SRC, _do_init)
- static void gst_data_repo_src_class_init (GstDataRepoSrcClass *klass)

  initialize the datareposrc's class
- static void gst_data_repo_src_init (GstDataRepoSrc *src)

  Initialize datareposrc.
- static gboolean gst_data_repo_src_parse_caps (GstDataRepoSrc *src, GstCaps *caps)

  Parse gst-caps to get media type and data size.
- static gboolean gst_data_repo_src_set_file_path (GstDataRepoSrc *src, const int prop, const gchar *file_path, GError **err)

  Function to set file path.
- static gboolean gst_data_repo_src_set_tensors_sequence (GstDataRepoSrc *src)

  Function to set tensors sequence.
- static guint64 gst_data_repo_src_get_file_offset (GstDataRepoSrc *src, guint sample_index)

  Function to get file offset with sample index.
- static void gst_data_repo_src_shuffle_samples_index (GstDataRepoSrc *src)

  Function to shuffle samples index.
- static gboolean gst_data_repo_src_epoch_is_done (GstDataRepoSrc *src)

  Function to check epoch and EOS.
- static GstFlowReturn gst_data_repo_src_read_tensors (GstDataRepoSrc *src, GstBuffer **buffer)

  Function to read tensors.
- static guint gst_data_repo_src_get_num_tensors (GstDataRepoSrc *src, guint shuffled_index)

  Function to get num_tensors from tensor_count_array.
- static GstFlowReturn gst_data_repo_src_read_flexible_or_sparse_tensors (GstDataRepoSrc *src, GstBuffer **buffer)

  Function to read flexible or sparse tensors.
- static gchar * gst_data_repo_src_get_image_filename (GstDataRepoSrc *src)

  Get image filename.
- static GstFlowReturn gst_data_repo_src_read_multi_images (GstDataRepoSrc *src, GstBuffer **buffer)

  Function to read multi image files.
- static GstFlowReturn gst_data_repo_src_read_others (GstDataRepoSrc *src, GstBuffer **buffer)

Function to read others media type (video, audio, octet and text)

- static gboolean gst_data_repo_src_start (GstDataRepoSrc *src)

  Start datareposrc, open the file.
- static void gst_data_repo_src_set_timestamp (GstDataRepoSrc *src, GstBuffer *buffer)

  Set timestamp.
- static gboolean gst_data_repo_get_caps_by_tensors_sequence (GstDataRepoSrc *src)

  Get caps with tensors_sequence applied.
- static gboolean gst_data_repo_src_read_json_file (GstDataRepoSrc *src)

  Read JSON file.

Variables

- static GstStaticPadTemplate srctemplate

9.5.1 Detailed Description

GStreamer plugin to read file in MLOps Data repository into buffers.

Date

31 January 2023

See also

https://github.com/nnstreamer/nnstreamer

Author

Hyunil Park hyunil46.park@samsung.com

Bug No known bugs except for NYI items

//! Example launch line |[gst-launch-1.0 datareposrc location=mnist.data json=mnist.json start-sample-index=3 stop-sample-index=202 epochs=5 ! \ ! tensor_sink gst-launch-1.0 datareposrc location=image_%02ld.png json=image.json start-sample-index=3 stop-sample-index=9 epochs=2 ! fakesink gst-launch-1.0 datareposrc location=audiofile json=audio.json ! fakesink gst-launch-1.0 datareposrc location=videofile json=video.json ! fakesink |] |[Unknown sample file(has not JSON) need to set caps and blocksize or set caps to tensors type without blocksize gst-launch-1.0 datareposrc blocksize=3176 location=unknown.data start-sample-index=3 stop-sample-index=202 epochs=5 \ caps ="application/octet-stream" ! tensor_converter input-dim=1:1:784:1,1:1:10:1 input-type=float32,float32 ! fakesink |] or |[gst-launch-1.0 datareposrc location=unknown.data start-sample-index=3 stop-sample-index=202 epochs=5 \ caps ="other/tensors, format=(string)static, framerate=(fraction)0/1, num_↩ tensors=(int)2, dimensions=(string)1:1:784:1.1:1:10:1, types=(string)float32.float32" \ ! fakesink]|

9.5.2 Macro Definition Documentation

9.5.2.1 _do_init

```
#define _do_init GST_DEBUG_CATEGORY_INIT (gst_data_repo_src_debug, "datareposrc", 0, "datareposrc
element");
```

Definition at line 108 of file gstdatareposrc.c.

9.5.2.2 DEFAULT_EPOCHS

```
#define DEFAULT_EPOCHS 1
```

Definition at line 91 of file gstdatareposrc.c.

9.5.2.3 DEFAULT_INDEX

```
#define DEFAULT_INDEX 0
```

Definition at line 90 of file gstdatareposrc.c.

9.5.2.4 DEFAULT_IS_SHUFFLE

```
#define DEFAULT_IS_SHUFFLE TRUE
```

Definition at line 92 of file gstdatareposrc.c.

9.5.2.5 GST_CAT_DEFAULT

```
#define GST_CAT_DEFAULT gst_data_repo_src_debug
```

Definition at line 74 of file gstdatareposrc.c.

9.5.2.6 gst_data_repo_src_parent_class

```
#define gst_data_repo_src_parent_class parent_class
```

Definition at line 111 of file gstdatareposrc.c.

9.5.2.7 O_BINARY

```
#define O_BINARY (0)
```

Definition at line 65 of file gstdatareposrc.c.

9.5.2.8 S_ISDIR

```
#define S_ISDIR(
              mode ) ((mode)&_S_IFDIR)
```

Definition at line 58 of file gstdatareposrc.c.

9.5.2.9 S_ISREG

```
#define S_ISREG(
              mode ) ((mode)&_S_IFREG)
```

Definition at line 55 of file gstdatareposrc.c.

9.5.2.10 S_ISSOCK

```
#define S_ISSOCK(
              x ) (0)
```

Definition at line 62 of file gstdatareposrc.c.

9.5.2.11 struct_stat

```
#define struct_stat struct stat
```

Definition at line 52 of file gstdatareposrc.c.

9.5.3 Enumeration Type Documentation

9.5.3.1 anonymous enum

```
anonymous enum
```

Enumerator

PROP_0	
PROP_LOCATION	
PROP_JSON	
PROP_START_SAMPLE_INDEX	
PROP_STOP_SAMPLE_INDEX	
PROP_EPOCHS	
PROP_IS_SHUFFLE	
PROP_TENSORS_SEQUENCE	
PROP_CAPS	

Definition at line 77 of file gstdatareposrc.c.

9.5.4 Function Documentation

9.5.4.1 G_DEFINE_TYPE_WITH_CODE()

```
G_DEFINE_TYPE_WITH_CODE (
            GstDataRepoSrc ,
            gst_data_repo_src ,
            GST_TYPE_PUSH_SRC ,
            _do_init  )
```

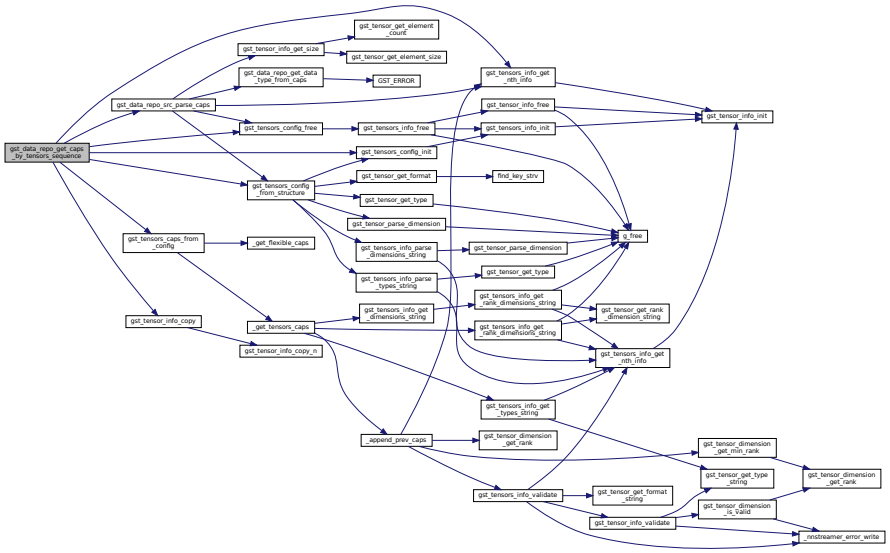
9.5.4.2 gst_data_repo_get_caps_by_tensors_sequence()

```
static gboolean gst_data_repo_get_caps_by_tensors_sequence (
            GstDataRepoSrc * src )  [static]
```

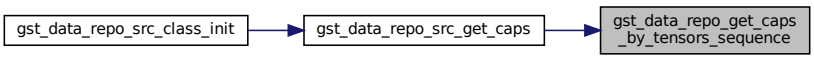
Get caps with tensors_sequence applied.

Definition at line 1326 of file gstdatareposrc.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.5.4.3 gst_data_repo_src_change_state()

```
static GstStateChangeReturn gst_data_repo_src_change_state (
            GstElement * element,
            GstStateChange transition )  [static]
```

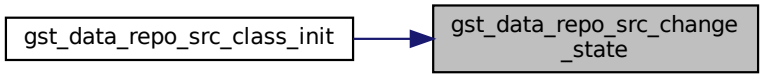
Change state of datareposrc.

if data_type is not GST_DATA_REPO_DATA_UNKNOWN and sample_size is 0 then 'caps' is set by property and sample size needs to be set by blocksize (in the case of otect and text)

A case of importing a sample format using 'caps' property without JSON.

Definition at line 1674 of file gstdatareposrc.c.

Here is the caller graph for this function:



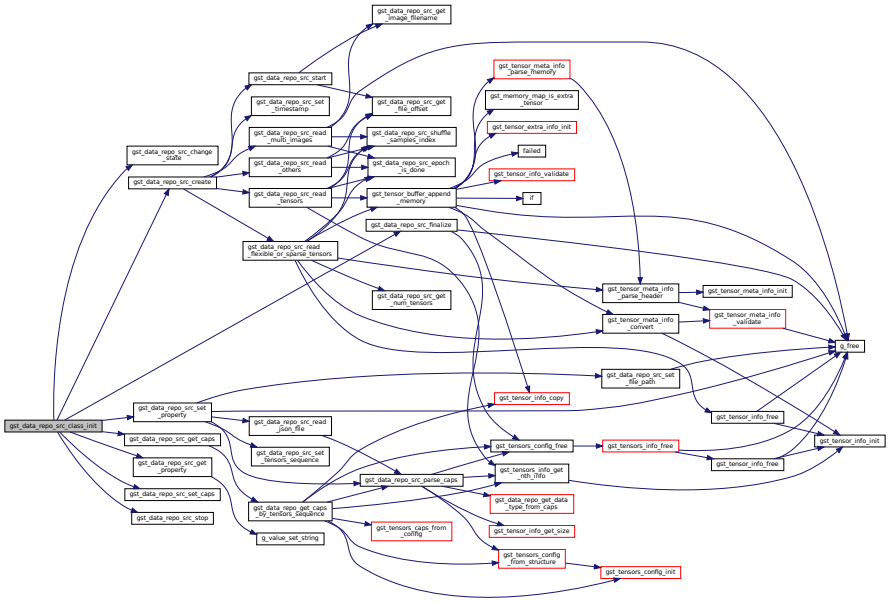
9.5.4.4 gst_data_repo_src_class_init()

```
static void gst_data_repo_src_class_init (
            GstDataRepoSrcClass * klass )  [static]
```

initialize the datareposrc's class

Definition at line 119 of file gstdatareposrc.c.

Here is the call graph for this function:



9.5.4.5 gst_data_repo_src_create()

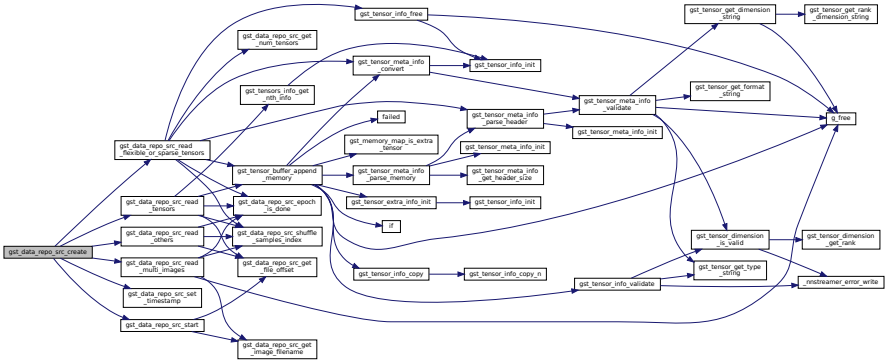
```
static GstFlowReturn gst_data_repo_src_create (
            GstPushSrc * pushsrc,
            GstBuffer ** buffer )  [static]
```

Function to create a buffer.

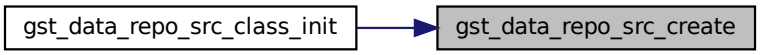
set_caps is completed after PAUSED_TO_PLAYING, so we cannot use change_state. datareposrc can get type and size after set_caps()

Definition at line 1263 of file gstdatareposrc.c.

Here is the call graph for this function:



Here is the caller graph for this function:



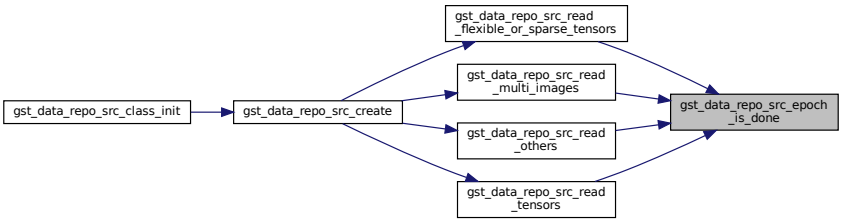
9.5.4.6 gst_data_repo_src_epoch_is_done()

```
static gboolean gst_data_repo_src_epoch_is_done (
            GstDataRepoSrc * src )  [static]
```

Function to check epoch and EOS.

Definition at line 553 of file gstdatareposrc.c.

Here is the caller graph for this function:



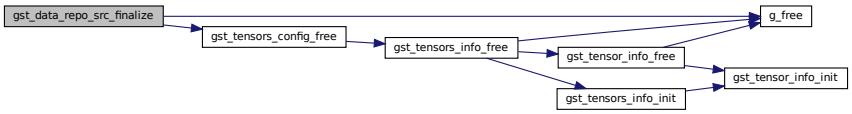
9.5.4.7 gst_data_repo_src_finalize()

```
static void gst_data_repo_src_finalize (
            GObject * object )  [static]
```

Function to finalize instance.

Definition at line 270 of file gstdatareposrc.c.

Here is the call graph for this function:



Here is the caller graph for this function:

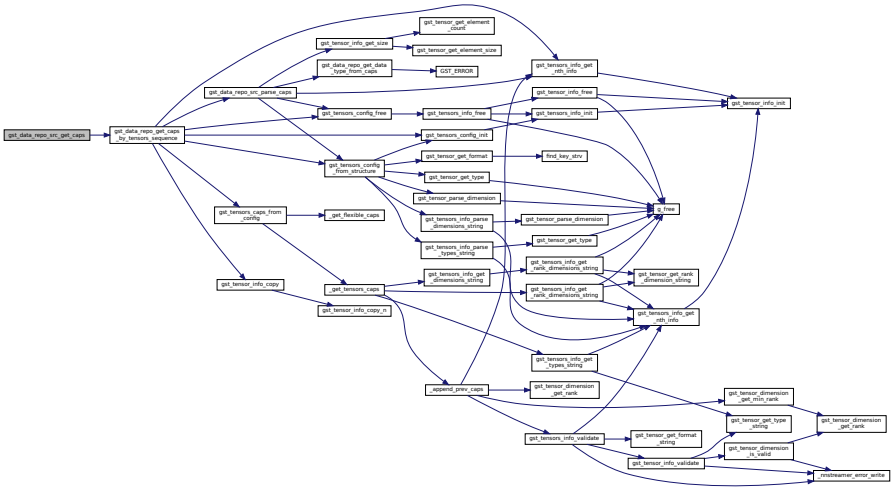
9.5.4.8 gst_data_repo_src_get_caps()

```
static GstCaps * gst_data_repo_src_get_caps (
            GstBaseSrc * basesrc,
            GstCaps * filter )  [static]
```

Get caps for caps negotiation.

Definition at line 1371 of file gstdatareposrc.c.

Here is the call graph for this function:



Here is the caller graph for this function:

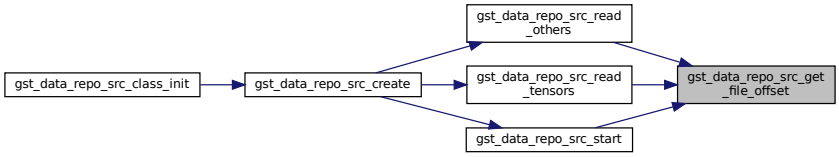
9.5.4.9 gst_data_repo_src_get_file_offset()

```
static guint64 gst_data_repo_src_get_file_offset (
            GstDataRepoSrc * src,
            guint sample_index )  [static]
```

Function to get file offset with sample index.

Definition at line 504 of file gstdatareposrc.c.

Here is the caller graph for this function:



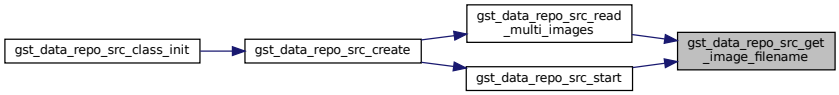
9.5.4.10 gst_data_repo_src_get_image_filename()

```
static gchar* gst_data_repo_src_get_image_filename (
            GstDataRepoSrc * src )  [static]
```

Get image filename.

Definition at line 894 of file gstdatareposrc.c.

Here is the caller graph for this function:



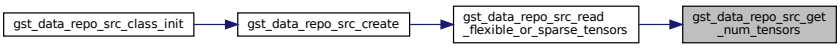
9.5.4.11 gst_data_repo_src_get_num_tensors()

```
static guint gst_data_repo_src_get_num_tensors (
            GstDataRepoSrc * src,
            guint shuffled_index )  [static]
```

Function to get num_tensors from tensor_count_array.

Definition at line 708 of file gstdatareposrc.c.

Here is the caller graph for this function:



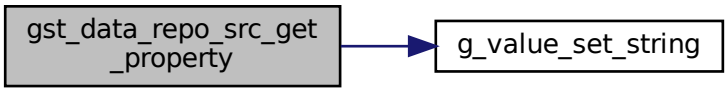
9.5.4.12 gst_data_repo_src_get_property()

```
static void gst_data_repo_src_get_property (
            GObject * object,
            guint prop_id,
            GValue * value,
            GParamSpec * pspec )  [static]
```

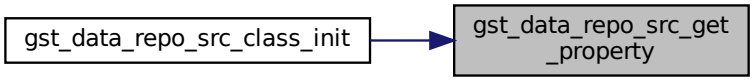
Getter datareposrc properties.

Definition at line 1630 of file gstdatareposrc.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.5.4.13 gst_data_repo_src_init()

```
static void gst_data_repo_src_init (
            GstDataRepoSrc * src )  [static]
```

Initialize datareposrc.

Definition at line 223 of file gstdatareposrc.c.

Here is the call graph for this function:

9.5.4.14 gst_data_repo_src_parse_caps()

```
static gboolean gst_data_repo_src_parse_caps (
            GstDataRepoSrc * src,
            GstCaps * caps )  [static]
```

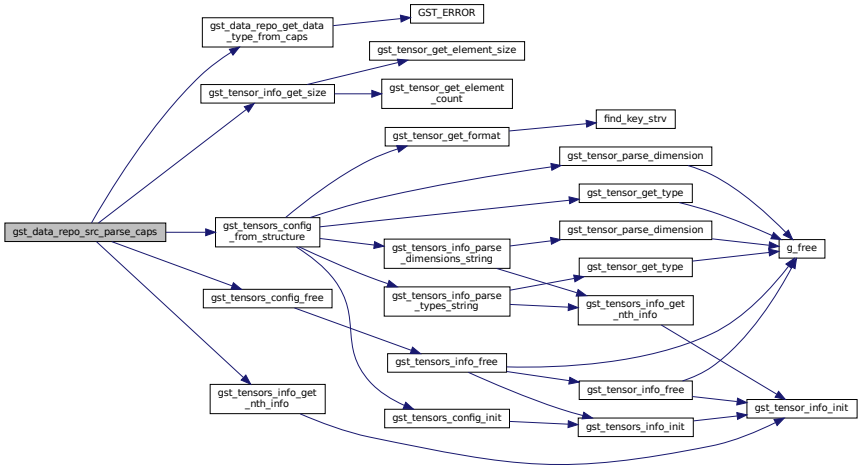
Parse gst-caps to get media type and data size.

Note

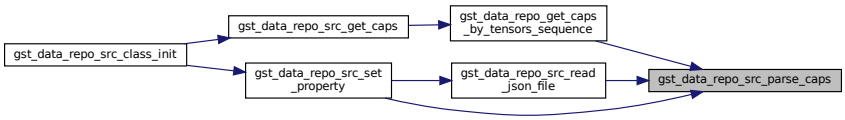
This function always takes the ownership of gst-caps.

Definition at line 303 of file gstdatareposrc.c.

Here is the call graph for this function:



Here is the caller graph for this function:



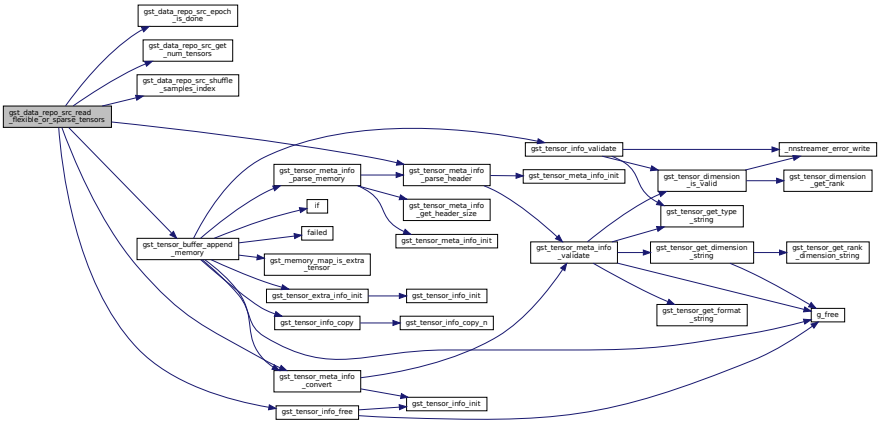
9.5.4.15 gst_data_repo_src_read_flexible_or_sparse_tensors()

```
static GstFlowReturn gst_data_repo_src_read_flexible_or_sparse_tensors (
            GstDataRepoSrc * src,
            GstBuffer ** buffer )  [static]
```

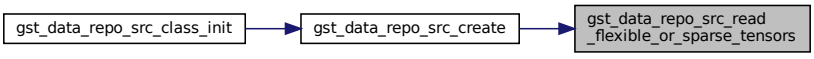
Function to read flexible or sparse tensors.

Definition at line 744 of file gstdatareposrc.c.

Here is the call graph for this function:



Here is the caller graph for this function:



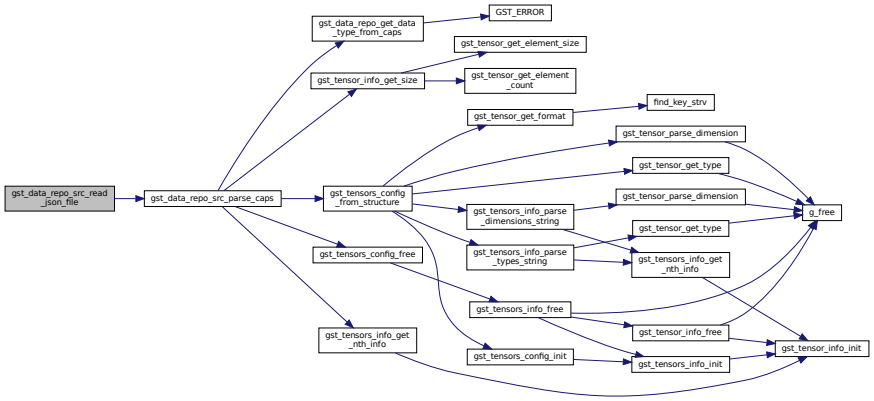
9.5.4.16 gst_data_repo_src_read_json_file()

```
static gboolean gst_data_repo_src_read_json_file (
            GstDataRepoSrc * src )  [static]
```

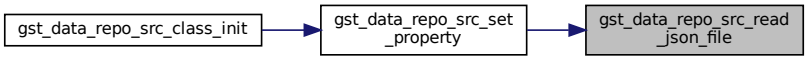
Read JSON file.

Definition at line 1414 of file gstdatareposrc.c.

Here is the call graph for this function:



Here is the caller graph for this function:



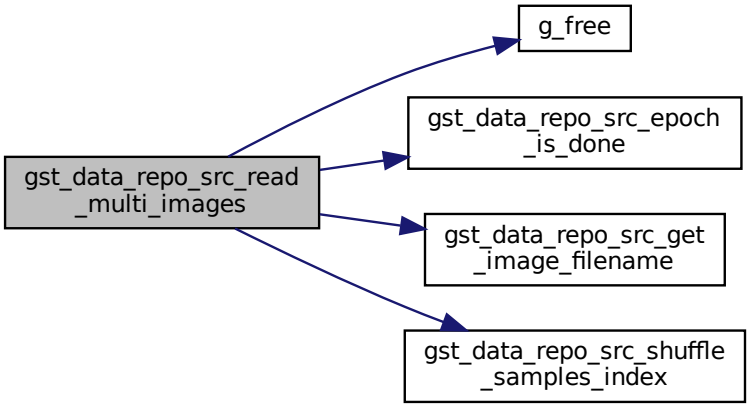
9.5.4.17 gst_data_repo_src_read_multi_images()

```
static GstFlowReturn gst_data_repo_src_read_multi_images (
            GstDataRepoSrc * src,
            GstBuffer ** buffer )  [static]
```

Function to read multi image files.

Definition at line 927 of file gstdatareposrc.c.

Here is the call graph for this function:



Here is the caller graph for this function:

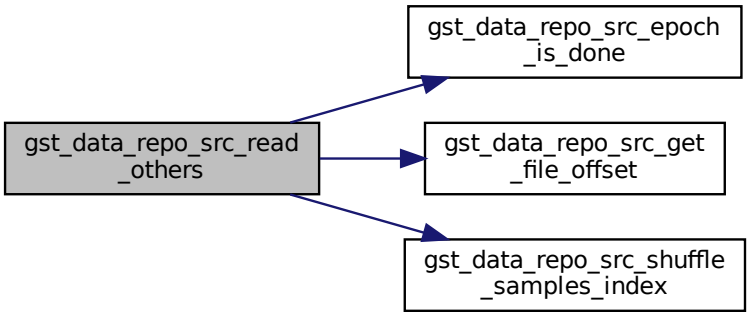
9.5.4.18 gst_data_repo_src_read_others()

```
static GstFlowReturn gst_data_repo_src_read_others (
            GstDataRepoSrc * src,
            GstBuffer ** buffer )  [static]
```

Function to read others media type (video, audio, octet and text)

Definition at line 1002 of file gstdatareposrc.c.

Here is the call graph for this function:



Here is the caller graph for this function:

9.5.4.19 gst_data_repo_src_read_tensors()

```
static GstFlowReturn gst_data_repo_src_read_tensors (
            GstDataRepoSrc * src,
            GstBuffer ** buffer )  [static]
```

Function to read tensors.

offset and sample_offset(byte size) are from 0. if the size of one sample is 6352 and num_tensors is 4, dimensions are '1:1:784:1' , '1:1:10:1', '1:1:784:1' and '1:1:10:1' with float32.

9.5.4.20 the offset of the second sample is as follows.

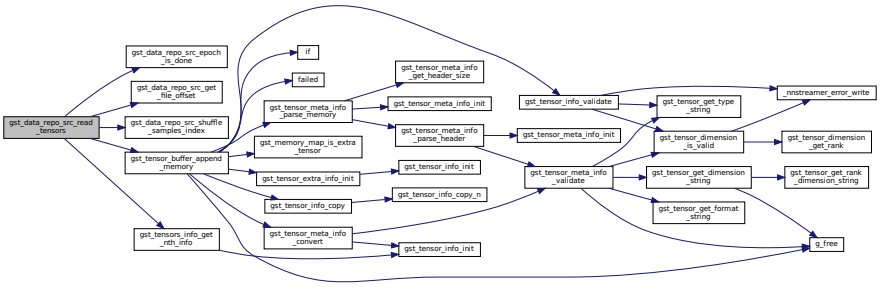
sample_offset: 6352 tensors index: [0 | 1 | 2 | 3] tensors_size: [3136 | 40 | 3136 | 40] tensors_offset: [0 | 3136 | 3176 | 6312]

9.5.4.21 fd offset: [6352 | 9488 | 9528 | 12664]

if user sets "tensor-sequence=2,1", datareposrc read offset 9528 then 9488.

Definition at line 570 of file gstdatareposrc.c.

Here is the call graph for this function:



Here is the caller graph for this function:

9.5.4.22 gst_data_repo_src_set_caps()

```
static gboolean gst_data_repo_src_set_caps (
            GstBaseSrc * basesrc,
            GstCaps * caps )  [static]
```

caps after caps negotiation

Definition at line 1401 of file gstdatareposrc.c.

Here is the caller graph for this function:

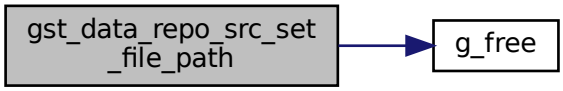
9.5.4.23 gst_data_repo_src_set_file_path()

```
static gboolean gst_data_repo_src_set_file_path (
            GstDataRepoSrc * src,
            const int prop,
            const gchar * file_path,
            GError ** err )  [static]
```

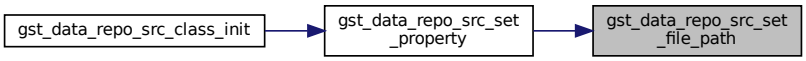
Function to set file path.

Definition at line 396 of file gstdatareposrc.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.5.4.24 gst_data_repo_src_set_property()

```
static void gst_data_repo_src_set_property (
            GObject * object,
            guint prop_id,
            const GValue * value,
            GParamSpec * pspec )  [static]
```

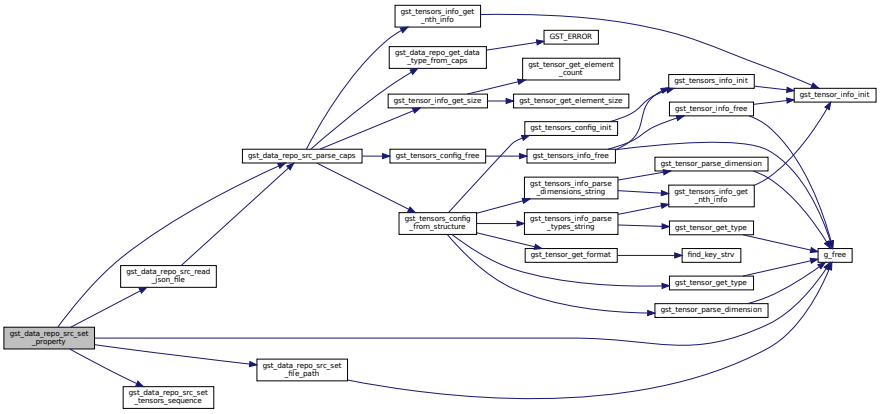
Setter for datareposrc properties.

To get caps, read JSON before Caps negotiation, to get information on sample data

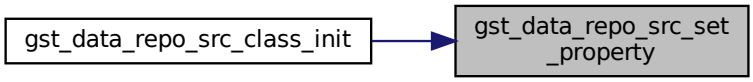
let's retry set tensors-sequence. if caps property is set later than tensors-sequence property, setting tensors-sequence fails because caps information is unknown.

Definition at line 1562 of file gstdatareposrc.c.

Here is the call graph for this function:



Here is the caller graph for this function:



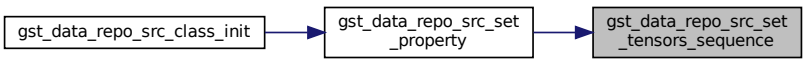
9.5.4.25 gst_data_repo_src_set_tensors_sequence()

```
static gboolean gst_data_repo_src_set_tensors_sequence (
            GstDataRepoSrc * src )  [static]
```

Function to set tensors sequence.

Definition at line 450 of file gstdatareposrc.c.

Here is the caller graph for this function:



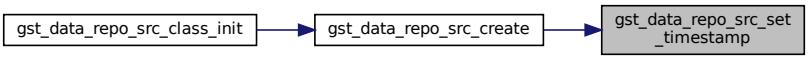
9.5.4.26 gst_data_repo_src_set_timestamp()

```
static void gst_data_repo_src_set_timestamp (
            GstDataRepoSrc * src,
            GstBuffer * buffer )  [static]
```

Set timestamp.

Definition at line 1225 of file gstdatareposrc.c.

Here is the caller graph for this function:



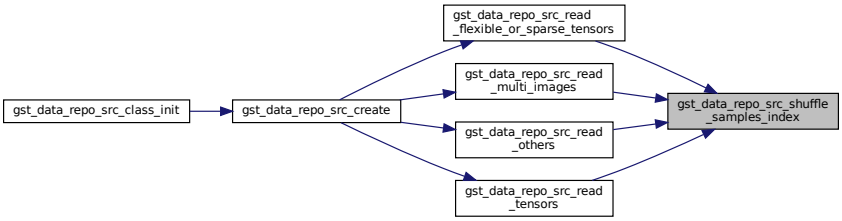
9.5.4.27 gst_data_repo_src_shuffle_samples_index()

```
static void gst_data_repo_src_shuffle_samples_index (
            GstDataRepoSrc * src )  [static]
```

Function to shuffle samples index.

Definition at line 521 of file gstdatareposrc.c.

Here is the caller graph for this function:



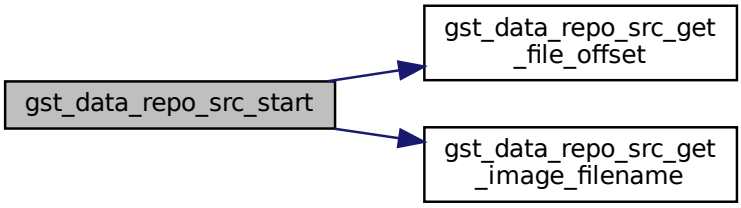
9.5.4.28 gst_data_repo_src_start()

```
static gboolean gst_data_repo_src_start (
            GstDataRepoSrc * src )  [static]
```

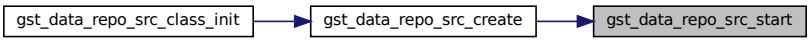
Start datareposrc, open the file.

Definition at line 1102 of file gstdatareposrc.c.

Here is the call graph for this function:



Here is the caller graph for this function:



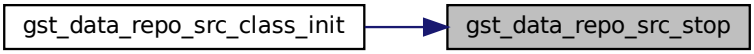
9.5.4.29 gst_data_repo_src_stop()

```
static gboolean gst_data_repo_src_stop (
            GstBaseSrc * basesrc )  [static]
```

Stop datareposrc, unmap and close the file.

Definition at line 1311 of file gstdatareposrc.c.

Here is the caller graph for this function:



9.5.4.30 GST_DEBUG_CATEGORY_STATIC()

```
GST_DEBUG_CATEGORY_STATIC (
            gst_data_repo_src_debug  )
```

9.5.5 Variable Documentation

9.5.5.1 srctemplate

```
GstStaticPadTemplate srctemplate  [static]
```

Initial value:

```
= GST_STATIC_PAD_TEMPLATE ("src",
    GST_PAD_SRC,
    GST_PAD_ALWAYS,
    GST_STATIC_CAPS_ANY)
```

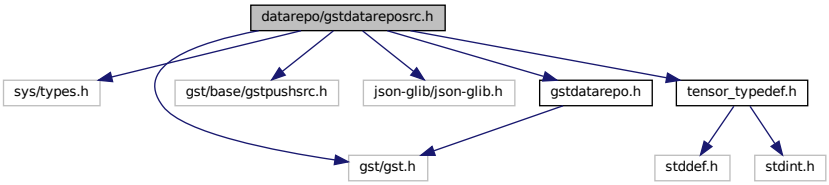
Definition at line 68 of file gstdatareposrc.c.

9.6 datarepo/gstdatareposrc.h File Reference

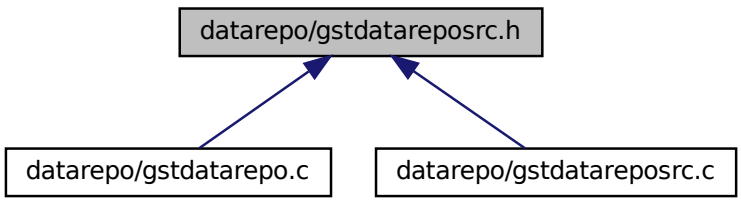
GStreamer plugin to read file in MLOps Data repository into buffers.

```
#include <sys/types.h>
#include <gst/gst.h>
#include <gst/base/gstpushsrc.h>
#include <json-glib/json-glib.h>
#include <tensor_typedef.h>
#include "gstdatarepo.h"
```

Include dependency graph for gstdatareposrc.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct _GstDataRepoSrc

  GstDataRepoSrc data structure.
- struct _GstDataRepoSrcClass

  GstDataRepoSrcClass data structure.

Macros

- #define GST_TYPE_DATA_REPO_SRC (gst_data_repo_src_get_type())
- #define GST_DATA_REPO_SRC(obj) (G_TYPE_CHECK_INSTANCE_CAST((obj),GST_TYPE_DATA_REPO_SRC,GstDataR
- #define GST_DATA_REPO_SRC_CLASS(klass) (G_TYPE_CHECK_CLASS_CAST((klass),GST_TYPE_DATA_REPO_SRC,
- #define GST_IS_DATA_REPO_SRC(obj) (G_TYPE_CHECK_INSTANCE_TYPE((obj),GST_TYPE_DATA_REPO_SRC))
- #define GST_IS_DATA_REPO_SRC_CLASS(klass) (G_TYPE_CHECK_CLASS_TYPE((klass),GST_TYPE_DATA_REPO_SR

Typedefs

- typedef struct _GstDataRepoSrc GstDataRepoSrc
- typedef struct _GstDataRepoSrcClass GstDataRepoSrcClass

Functions

- GType gst_data_repo_src_get_type (void)

9.6.1 Detailed Description

GStreamer plugin to read file in MLOps Data repository into buffers.

Date

31 January 2023

See also

https://github.com/nnstreamer/nnstreamer

Author

Hyunil Park hyunil46.park@samsung.com

Bug No known bugs except for NYI items

9.6.2 Macro Definition Documentation

9.6.2.1 GST_DATA_REPO_SRC

```
#define GST_DATA_REPO_SRC(
              obj ) (G_TYPE_CHECK_INSTANCE_CAST((obj),GST_TYPE_DATA_REPO_SRC,GstDataRepoSrc))
```

Definition at line 26 of file gstdatareposrc.h.

9.6.2.2 GST_DATA_REPO_SRC_CLASS

```
#define GST_DATA_REPO_SRC_CLASS(
              klass ) (G_TYPE_CHECK_CLASS_CAST((klass),GST_TYPE_DATA_REPO_SRC,GstDataRepoSrcClass))
```

Definition at line 28 of file gstdatareposrc.h.

9.6.2.3 GST_IS_DATA_REPO_SRC

```
#define GST_IS_DATA_REPO_SRC(
              obj ) (G_TYPE_CHECK_INSTANCE_TYPE((obj),GST_TYPE_DATA_REPO_SRC))
```

Definition at line 30 of file gstdatareposrc.h.

9.6.2.4 GST_IS_DATA_REPO_SRC_CLASS

```
#define GST_IS_DATA_REPO_SRC_CLASS(
              klass ) (G_TYPE_CHECK_CLASS_TYPE((klass),GST_TYPE_DATA_REPO_SRC))
```

Definition at line 32 of file gstdatareposrc.h.

9.6.2.5 GST_TYPE_DATA_REPO_SRC

```
#define GST_TYPE_DATA_REPO_SRC (gst_data_repo_src_get_type())
```

Definition at line 24 of file gstdatareposrc.h.

9.6.3 Typedef Documentation

9.6.3.1 GstDataRepoSrc

```
typedef struct _GstDataRepoSrc GstDataRepoSrc
```

Definition at line 35 of file gstdatareposrc.h.

9.6.3.2 GstDataRepoSrcClass

```
typedef struct _GstDataRepoSrcClass GstDataRepoSrcClass
```

Definition at line 36 of file gstdatareposrc.h.

9.6.4 Function Documentation

9.6.4.1 gst_data_repo_src_get_type()

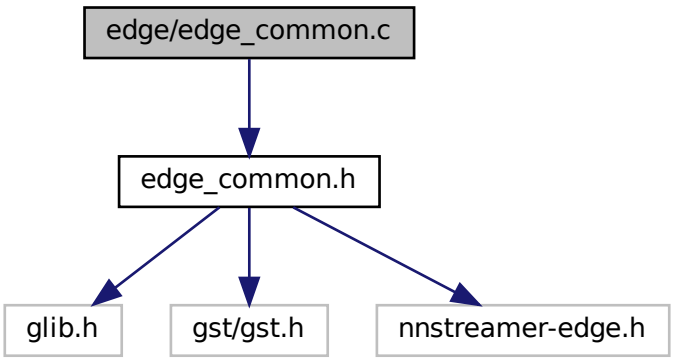
```
GType gst_data_repo_src_get_type (
            void  )
```

9.7 edge/edge_common.c File Reference

Common functions for edge sink and src.

```
#include "edge_common.h"
```

Include dependency graph for edge_common.c:



Functions

- GType gst_edge_get_connect_type (void)

  register GEnumValue array for edge protocol property handling

9.7.1 Detailed Description

Common functions for edge sink and src.

Date

01 Aug 2022

Author

Yechan Choi yechan9.choi@samsung.com

See also

http://github.com/nnstreamer/nnstreamer

Bug No known bugs

9.7.2 Function Documentation

9.7.2.1 gst_edge_get_connect_type()

```
GType gst_edge_get_connect_type (
            void  )
```

register GEnumValue array for edge protocol property handling

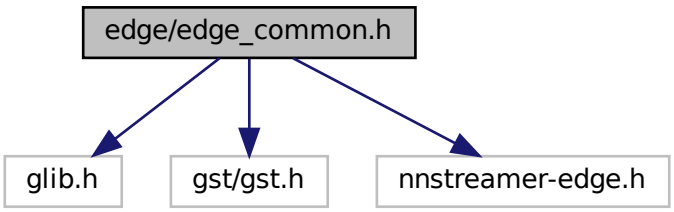
Definition at line 23 of file edge_common.c.

9.8 edge/edge_common.h File Reference

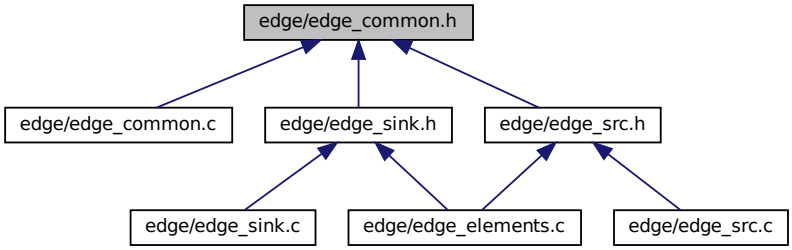
Common functions for edge sink and src.

```
#include <glib.h>
#include <gst/gst.h>
#include <nnstreamer-edge.h>
```

Include dependency graph for edge_common.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define GST_EDGE_PACKAGE "GStreamer Edge Plugins"
- #define GST_EDGE_ELEM_NAME_SINK "edgesink"
- #define GST_EDGE_ELEM_NAME_SRC "edgesrc"
- #define DEFAULT_HOST "localhost"
- #define DEFAULT_PORT 3000
- #define DEFAULT_CONNECT_TYPE (NNS_EDGE_CONNECT_TYPE_TCP)
- #define GST_TYPE_EDGE_CONNECT_TYPE (gst_edge_get_connect_type ())

Functions

- G_BEGIN_DECLS GType gst_edge_get_connect_type (void)

  register GEnumValue array for edge protocol property handling

9.8.1 Detailed Description

Common functions for edge sink and src.

Date

01 Aug 2022

Author

Yechan Choi yechan9.choi@samsung.com

See also

http://github.com/nnstreamer/nnstreamer

Bug No known bugs

9.8.2 Macro Definition Documentation

9.8.2.1 DEFAULT_CONNECT_TYPE

```
#define DEFAULT_CONNECT_TYPE (NNS_EDGE_CONNECT_TYPE_TCP)
```

Definition at line 27 of file edge_common.h.

9.8.2.2 DEFAULT_HOST

```
#define DEFAULT_HOST "localhost"
```

Definition at line 25 of file edge_common.h.

9.8.2.3 DEFAULT_PORT

```
#define DEFAULT_PORT 3000
```

Definition at line 26 of file edge_common.h.

9.8.2.4 GST_EDGE_ELEM_NAME_SINK

```
#define GST_EDGE_ELEM_NAME_SINK "edgesink"
```

Definition at line 23 of file edge_common.h.

9.8.2.5 GST_EDGE_ELEM_NAME_SRC

```
#define GST_EDGE_ELEM_NAME_SRC "edgesrc"
```

Definition at line 24 of file edge_common.h.

9.8.2.6 GST_EDGE_PACKAGE

```
#define GST_EDGE_PACKAGE "GStreamer Edge Plugins"
```

Definition at line 21 of file edge_common.h.

9.8.2.7 GST_TYPE_EDGE_CONNECT_TYPE

```
#define GST_TYPE_EDGE_CONNECT_TYPE (gst_edge_get_connect_type ())
```

Definition at line 28 of file edge_common.h.

9.8.3 Function Documentation

9.8.3.1 gst_edge_get_connect_type()

```
G_BEGIN_DECLS GType gst_edge_get_connect_type (
            void  )
```

register GEnumValue array for edge protocol property handling

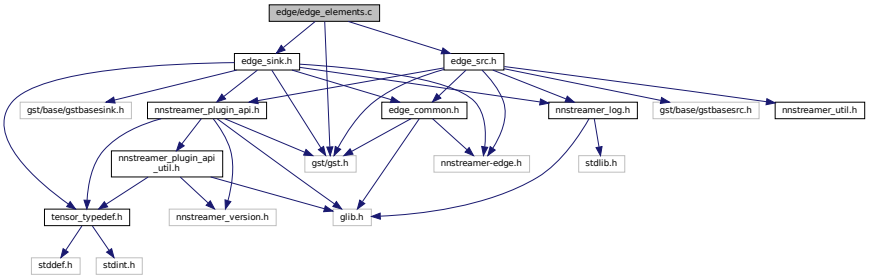
Definition at line 23 of file edge_common.c.

9.9 edge/edge_elements.c File Reference

Register edge plugins.

```
#include <gst/gst.h>
#include "edge_sink.h"
#include "edge_src.h"
```

Include dependency graph for edge_elements.c:



Macros

- #define PACKAGE GST_EDGE_PACKAGE

Functions

- static gboolean plugin_init (GstPlugin *plugin)

  The entry point of the Gstreamer Edge plugin.

9.9.1 Detailed Description

Register edge plugins.

Copyright (C) 2022 Samsung Electronics Co., Ltd.

Date

02 Aug 2022

Author

Yechan Choi yechan9.choi@samsung.com

See also

http://github.com/nnstreamer/nnstreamer

Bug No known bugs

9.9.2 Macro Definition Documentation

9.9.2.1 PACKAGE

```
#define PACKAGE GST_EDGE_PACKAGE
```

Definition at line 38 of file edge_elements.c.

9.9.3 Function Documentation

9.9.3.1 plugin_init()

```
static gboolean plugin_init (
            GstPlugin * plugin )  [static]
```

The entry point of the Gstreamer Edge plugin.

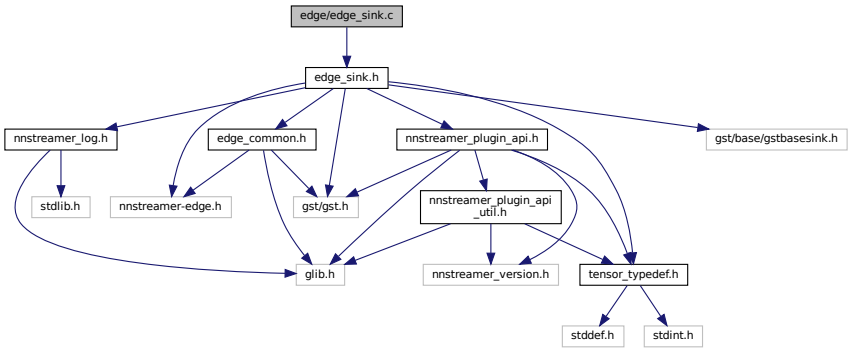
Definition at line 22 of file edge_elements.c.

9.10 edge/edge_sink.c File Reference

Publish incoming streams.

```
#include "edge_sink.h"
```

Include dependency graph for edge_sink.c:



Macros

- #define GST_CAT_DEFAULT gst_edgesink_debug
- #define DEFAULT_MQTT_HOST "127.0.0.1"
- #define DEFAULT_MQTT_PORT 1883
- #define gst_edgesink_parent_class parent_class

Enumerations

- enum {
  PROP_0, PROP_HOST, PROP_PORT, PROP_DEST_HOST,
  PROP_DEST_PORT, PROP_CONNECT_TYPE, PROP_TOPIC, PROP_WAIT_CONNECTION,
  PROP_CONNECTION_TIMEOUT, PROP_CUSTOM_LIB, PROP_LAST }

  edgesink properties

Functions

- GST_DEBUG_CATEGORY_STATIC (gst_edgesink_debug)
- G_DEFINE_TYPE (GstEdgeSink, gst_edgesink, GST_TYPE_BASE_SINK)
- static void gst_edgesink_set_property (GObject *object, guint prop_id, const GValue *value, GParamSpec *pspec)

  set property
- static void gst_edgesink_get_property (GObject *object, guint prop_id, GValue *value, GParamSpec *pspec)

  get property
- static void gst_edgesink_finalize (GObject *object)

  finalize the object
- static gboolean gst_edgesink_start (GstBaseSink *basesink)

  start processing of edgesink
- static gboolean gst_edgesink_stop (GstBaseSink *basesink)

  Stop processing of edgesink.
- static GstFlowReturn gst_edgesink_render (GstBaseSink *basesink, GstBuffer *buffer)

  render buffer, send buffer
- static gboolean gst_edgesink_set_caps (GstBaseSink *basesink, GstCaps *caps)

  An implementation of the set_caps vmethod in GstBaseSinkClass.
- static gchar * gst_edgesink_get_host (GstEdgeSink *self)

  getter for the 'host' property.
- static void gst_edgesink_set_host (GstEdgeSink *self, const gchar *host)

  setter for the 'host' property.
- static guint16 gst_edgesink_get_port (GstEdgeSink *self)

  getter for the 'port' property.
- static void gst_edgesink_set_port (GstEdgeSink *self, const guint16 port)

  setter for the 'port' property.
- static nns_edge_connect_type_e gst_edgesink_get_connect_type (GstEdgeSink *self)

  getter for the 'connect_type' property.
- static void gst_edgesink_set_connect_type (GstEdgeSink *self, const nns_edge_connect_type_e connect↩_type)

  setter for the 'connect_type' property.
- static void gst_edgesink_class_init (GstEdgeSinkClass *klass)

  initialize the class
- static void gst_edgesink_init (GstEdgeSink *self)

  initialize the new element
- static int _nns_edge_event_cb (nns_edge_event_h event_h, void *user_data)

  nnstreamer-edge event callback.
- static gboolean _wait_connection (GstEdgeSink *sink)

  If wait-connection is enabled, wait for connection until the connection is established or timeout occurs. Otherwise, return immediately.

Variables

- static GstStaticPadTemplate sinktemplate

  the capabilities of the inputs.

9.10.1 Detailed Description

Publish incoming streams.

Copyright (C) 2022 Samsung Electronics Co., Ltd.

Date

01 Aug 2022

Author

Yechan Choi yechan9.choi@samsung.com

See also

http://github.com/nnstreamer/nnstreamer

Bug No known bugs

9.10.2 Macro Definition Documentation

9.10.2.1 DEFAULT_MQTT_HOST

```
#define DEFAULT_MQTT_HOST "127.0.0.1"
```

Definition at line 49 of file edge_sink.c.

9.10.2.2 DEFAULT_MQTT_PORT

```
#define DEFAULT_MQTT_PORT 1883
```

Definition at line 50 of file edge_sink.c.

9.10.2.3 GST_CAT_DEFAULT

```
#define GST_CAT_DEFAULT gst_edgesink_debug
```

Definition at line 20 of file edge_sink.c.

9.10.2.4 gst_edgesink_parent_class

```
#define gst_edgesink_parent_class parent_class
```

Definition at line 52 of file edge_sink.c.

9.10.3 Enumeration Type Documentation

9.10.3.1 anonymous enum

```
anonymous enum
```

edgesink properties

Enumerator

PROP_0	
PROP_HOST	
PROP_PORT	
PROP_DEST_HOST	
PROP_DEST_PORT	
PROP_CONNECT_TYPE	
PROP_TOPIC	
PROP_WAIT_CONNECTION	
PROP_CONNECTION_TIMEOUT	
PROP_CUSTOM_LIB	
PROP_LAST	

Definition at line 33 of file edge_sink.c.

9.10.4 Function Documentation

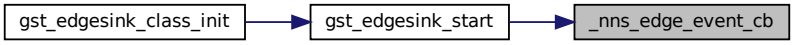
9.10.4.1 _nns_edge_event_cb()

```
static int _nns_edge_event_cb (
             nns_edge_event_h event_h,
             void * user_data )  [static]
```

nnstreamer-edge event callback.

Definition at line 306 of file edge_sink.c.

Here is the caller graph for this function:



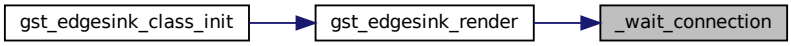
9.10.4.2 _wait_connection()

```
static gboolean _wait_connection (
             GstEdgeSink * sink )  [static]
```

If wait-connection is enabled, wait for connection until the connection is established or timeout occurs. Otherwise, return immediately.

Definition at line 400 of file edge_sink.c.

Here is the caller graph for this function:



9.10.4.3 G_DEFINE_TYPE()

```
G_DEFINE_TYPE (
             GstEdgeSink ,
             gst_edgesink ,
             GST_TYPE_BASE_SINK  )
```

9.10.4.4 GST_DEBUG_CATEGORY_STATIC()

```
GST_DEBUG_CATEGORY_STATIC (
            gst_edgesink_debug  )
```

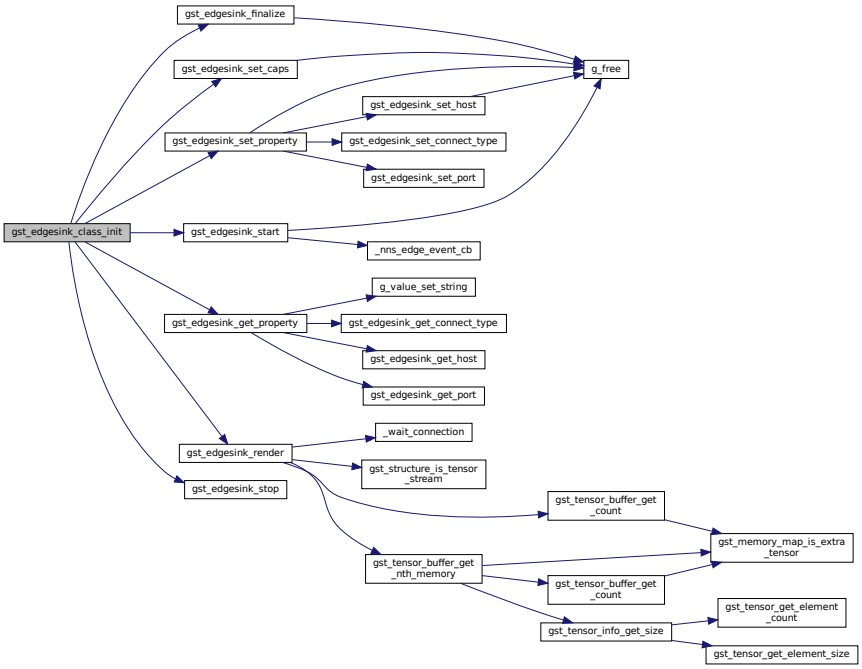
9.10.4.5 gst_edgesink_class_init()

```
static void gst_edgesink_class_init (
            GstEdgeSinkClass * klass )  [static]
```

initialize the class

Definition at line 84 of file edge_sink.c.

Here is the call graph for this function:



9.10.4.6 gst_edgesink_finalize()

```
static void gst_edgesink_finalize (
            GObject * object )  [static]
```

finalize the object

Definition at line 275 of file edge_sink.c.

Here is the call graph for this function:

Here is the caller graph for this function:

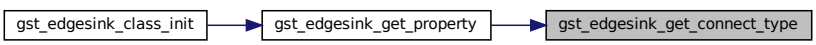
9.10.4.7 gst_edgesink_get_connect_type()

```
static nns_edge_connect_type_e gst_edgesink_get_connect_type (
            GstEdgeSink * self )  [static]
```

getter for the 'connect_type' property.

Definition at line 589 of file edge_sink.c.

Here is the caller graph for this function:



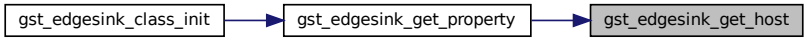
9.10.4.8 gst_edgesink_get_host()

```
static gchar * gst_edgesink_get_host (
            GstEdgeSink * self )  [static]
```

getter for the 'host' property.

Definition at line 551 of file edge_sink.c.

Here is the caller graph for this function:



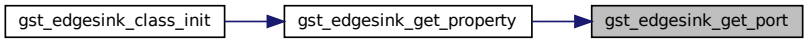
9.10.4.9 gst_edgesink_get_port()

```
static guint16 gst_edgesink_get_port (
            GstEdgeSink * self )  [static]
```

getter for the 'port' property.

Definition at line 571 of file edge_sink.c.

Here is the caller graph for this function:



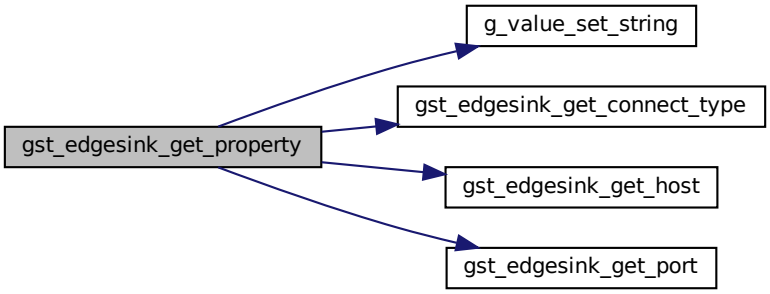
9.10.4.10 gst_edgesink_get_property()

```
static void gst_edgesink_get_property (
            GObject * object,
            guint prop_id,
            GValue * value,
            GParamSpec * pspec )  [static]
```

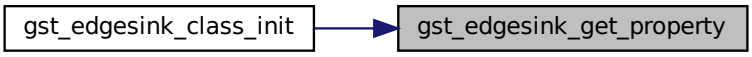
get property

Definition at line 232 of file edge_sink.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.10.4.11 gst_edgesink_init()

```
static void gst_edgesink_init (
             GstEdgeSink * self )  [static]
```

initialize the new element

Definition at line 158 of file edge_sink.c.

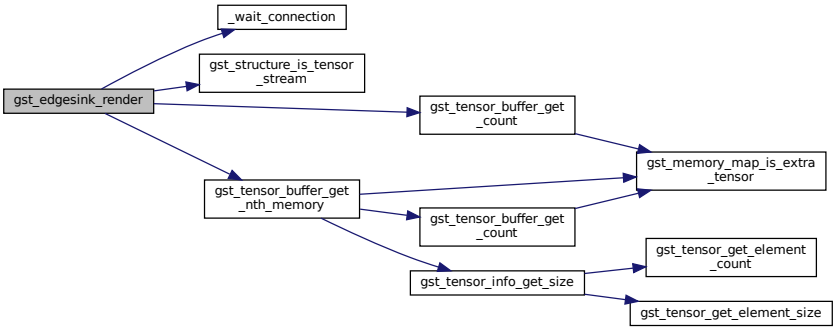
9.10.4.12 gst_edgesink_render()

```
static GstFlowReturn gst_edgesink_render (
             GstBaseSink * basesink,
             GstBuffer * buffer )  [static]
```

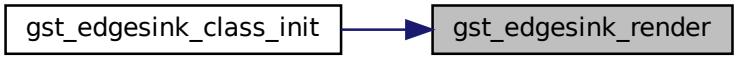
render buffer, send buffer

Definition at line 450 of file edge_sink.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.10.4.13 gst_edgesink_set_caps()

```
static gboolean gst_edgesink_set_caps (
            GstBaseSink * basesink,
            GstCaps * caps )  [static]
```

An implementation of the set_caps vmethod in GstBaseSinkClass.

Definition at line 524 of file edge_sink.c.

Here is the call graph for this function:

Here is the caller graph for this function:

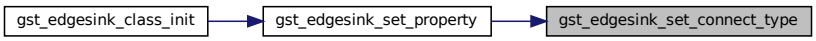
9.10.4.14 gst_edgesink_set_connect_type()

```
static void gst_edgesink_set_connect_type (
            GstEdgeSink * self,
            const nns_edge_connect_type_e connect_type )  [static]
```

setter for the 'connect_type' property.

Definition at line 598 of file edge_sink.c.

Here is the caller graph for this function:



9.10.4.15 gst_edgesink_set_host()

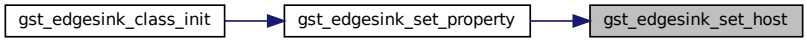
```
static void gst_edgesink_set_host (
            GstEdgeSink * self,
            const gchar * host )  [static]
```

setter for the 'host' property.

Definition at line 560 of file edge_sink.c.

Here is the call graph for this function:

Here is the caller graph for this function:



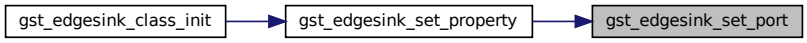
9.10.4.16 gst_edgesink_set_port()

```
static void gst_edgesink_set_port (
            GstEdgeSink * self,
            const guint16 port )  [static]
```

setter for the 'port' property.

Definition at line 580 of file edge_sink.c.

Here is the caller graph for this function:



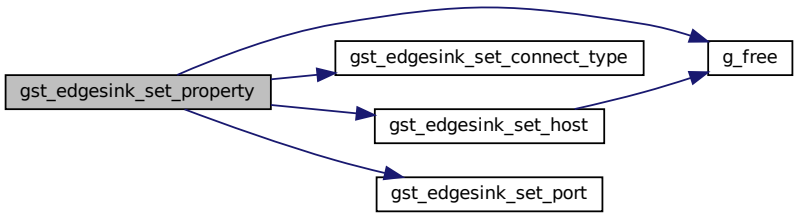
9.10.4.17 gst_edgesink_set_property()

```
static void gst_edgesink_set_property (
            GObject * object,
            guint prop_id,
            const GValue * value,
            GParamSpec * pspec )  [static]
```

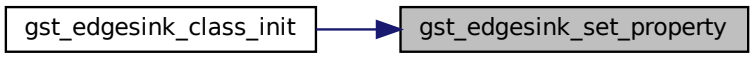
set property

Definition at line 178 of file edge_sink.c.

Here is the call graph for this function:



Here is the caller graph for this function:



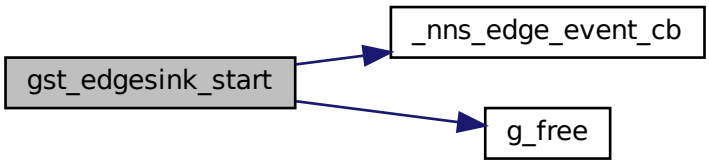
9.10.4.18 gst_edgesink_start()

```
static gboolean gst_edgesink_start (
            GstBaseSink * basesink )  [static]
```

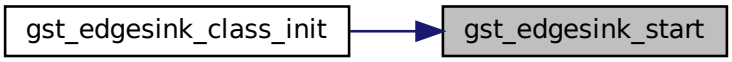
start processing of edgesink

Definition at line 338 of file edge_sink.c.

Here is the call graph for this function:



Here is the caller graph for this function:



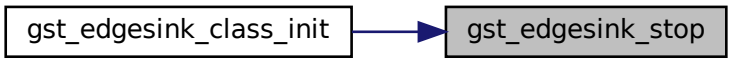
9.10.4.19 gst_edgesink_stop()

```
static gboolean gst_edgesink_stop (
            GstBaseSink * basesink )  [static]
```

Stop processing of edgesink.

Definition at line 432 of file edge_sink.c.

Here is the caller graph for this function:



9.10.5 Variable Documentation

9.10.5.1 sinktemplate

```
GstStaticPadTemplate sinktemplate  [static]
```

Initial value:

```
= GST_STATIC_PAD_TEMPLATE ("sink",
    GST_PAD_SINK,
    GST_PAD_ALWAYS,
    GST_STATIC_CAPS_ANY)
```

the capabilities of the inputs.

Definition at line 25 of file edge_sink.c.

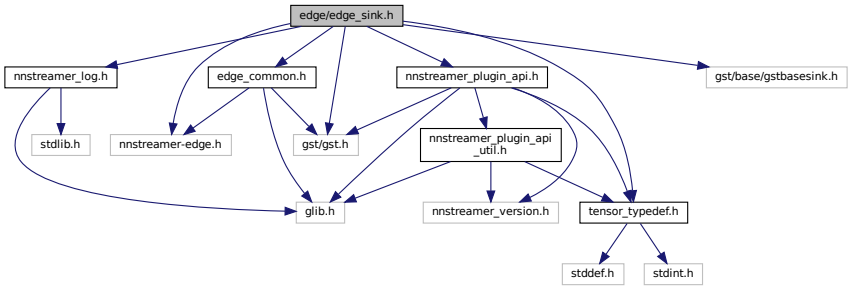
9.11 edge/edge_sink.h File Reference

Publish incoming streams.

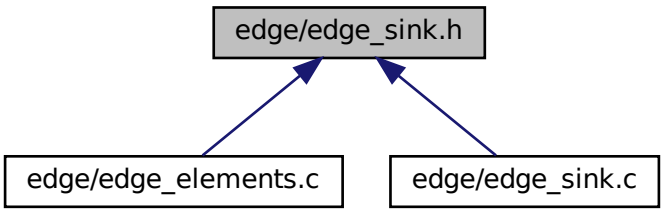
```
#include <gst/gst.h>
#include <gst/base/gstbasesink.h>
#include "edge_common.h"
#include <nnstreamer-edge.h>
#include "nnstreamer_log.h"
#include "nnstreamer_plugin_api.h"
```

```
#include "tensor_typedef.h"
```

Include dependency graph for edge_sink.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct _GstEdgeSink

  GstEdgeSink data structure.
- struct _GstEdgeSinkClass

  GstEdgeSinkClass data structure.

Macros

- #define GST_TYPE_EDGESINK (gst_edgesink_get_type())
- #define GST_EDGESINK(obj) (G_TYPE_CHECK_INSTANCE_CAST((obj),GST_TYPE_EDGESINK,GstEdgeSink))
- #define GST_EDGESINK_CLASS(klass) (G_TYPE_CHECK_CLASS_CAST((klass),GST_TYPE_EDGESINK,GstEdgeSinkCla
- #define GST_IS_EDGESINK(obj) (G_TYPE_CHECK_INSTANCE_TYPE((obj),GST_TYPE_EDGESINK))
- #define GST_IS_EDGESINK_CLASS(klass) (G_TYPE_CHECK_CLASS_TYPE((klass),GST_TYPE_EDGESINK))
- #define GST_EDGESINK_CAST(obj) ((GstEdgeSink *)(obj))

Typedefs

- typedef struct _GstEdgeSink GstEdgeSink
- typedef struct _GstEdgeSinkClass GstEdgeSinkClass

Functions

- GType gst_edgesink_get_type (void)

9.11.1 Detailed Description

Publish incoming streams.

Date

01 Aug 2022

Author

Yechan Choi yechan9.choi@samsung.com

See also

http://github.com/nnstreamer/nnstreamer

Bug No known bugs

9.11.2 Macro Definition Documentation

9.11.2.1 GST_EDGESINK

```
#define GST_EDGESINK(
              obj ) (G_TYPE_CHECK_INSTANCE_CAST((obj),GST_TYPE_EDGESINK,GstEdgeSink))
```

Definition at line 27 of file edge_sink.h.

9.11.2.2 GST_EDGESINK_CAST

```
#define GST_EDGESINK_CAST(
              obj ) ((GstEdgeSink *)(obj))
```

Definition at line 35 of file edge_sink.h.

9.11.2.3 GST_EDGESINK_CLASS

```
#define GST_EDGESINK_CLASS(
              klass ) (G_TYPE_CHECK_CLASS_CAST((klass),GST_TYPE_EDGESINK,GstEdgeSinkClass))
```

Definition at line 29 of file edge_sink.h.

9.11.2.4 GST_IS_EDGESINK

```
#define GST_IS_EDGESINK(
              obj ) (G_TYPE_CHECK_INSTANCE_TYPE((obj),GST_TYPE_EDGESINK))
```

Definition at line 31 of file edge_sink.h.

9.11.2.5 GST_IS_EDGESINK_CLASS

```
#define GST_IS_EDGESINK_CLASS(
              klass ) (G_TYPE_CHECK_CLASS_TYPE((klass),GST_TYPE_EDGESINK))
```

Definition at line 33 of file edge_sink.h.

9.11.2.6 GST_TYPE_EDGESINK

```
#define GST_TYPE_EDGESINK (gst_edgesink_get_type())
```

Definition at line 25 of file edge_sink.h.

9.11.3 Typedef Documentation

9.11.3.1 GstEdgeSink

```
typedef struct _GstEdgeSink GstEdgeSink
```

Definition at line 36 of file edge_sink.h.

9.11.3.2 GstEdgeSinkClass

```
typedef struct _GstEdgeSinkClass GstEdgeSinkClass
```

Definition at line 37 of file edge_sink.h.

9.11.4 Function Documentation

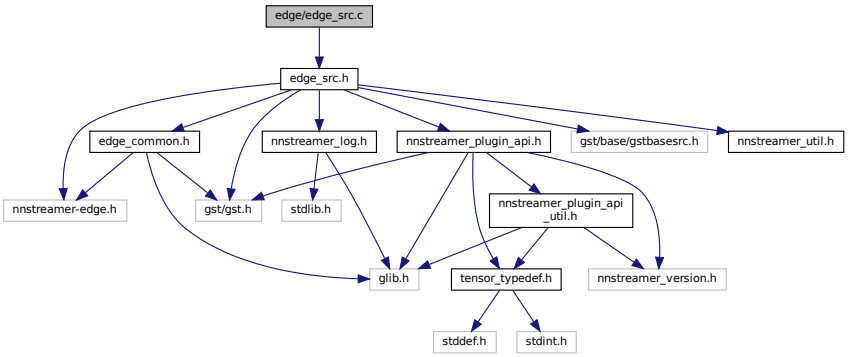
9.11.4.1 gst_edgesink_get_type()

```
GType gst_edgesink_get_type (
            void  )
```

9.12 edge/edge_src.c File Reference

Subscribe and push incoming data to the GStreamer pipeline.

```
#include "edge_src.h"
```
Include dependency graph for edge_src.c:



Macros

- #define GST_CAT_DEFAULT gst_edgesrc_debug
- #define gst_edgesrc_parent_class parent_class

Enumerations

- enum {
  PROP_0, PROP_HOST, PROP_PORT, PROP_DEST_HOST,
  PROP_DEST_PORT, PROP_CONNECT_TYPE, PROP_TOPIC, PROP_CUSTOM_LIB,
  PROP_LAST }

  edgesrc properties

Functions

- GST_DEBUG_CATEGORY_STATIC (gst_edgesrc_debug)
- G_DEFINE_TYPE (GstEdgeSrc, gst_edgesrc, GST_TYPE_BASE_SRC)
- static void gst_edgesrc_set_property (GObject *object, guint prop_id, const GValue *value, GParamSpec *pspec)

  set property
- static void gst_edgesrc_get_property (GObject *object, guint prop_id, GValue *value, GParamSpec *pspec)

  get property
- static void gst_edgesrc_class_finalize (GObject *object)

  finalize the object
- static gboolean gst_edgesrc_start (GstBaseSrc *basesrc)

  start edgesrc, called when state changed null to ready
- static gboolean gst_edgesrc_stop (GstBaseSrc *basesrc)

  Stop edgesrc, called when state changed ready to null.
- static GstFlowReturn gst_edgesrc_create (GstBaseSrc *basesrc, guint64 offset, guint size, GstBuffer **out_buf)

  Create a buffer containing the subscribed data.
- static gchar * gst_edgesrc_get_dest_host (GstEdgeSrc *self)

  getter for the 'host' property.
- static void gst_edgesrc_set_dest_host (GstEdgeSrc *self, const gchar *dest_host)

  setter for the 'host' property.
- static guint16 gst_edgesrc_get_dest_port (GstEdgeSrc *self)

  getter for the 'port' property.
- static void gst_edgesrc_set_dest_port (GstEdgeSrc *self, const guint16 dest_port)

  setter for the 'port' property.
- static nns_edge_connect_type_e gst_edgesrc_get_connect_type (GstEdgeSrc *self)

  getter for the 'connect_type' property.
- static void gst_edgesrc_set_connect_type (GstEdgeSrc *self, const nns_edge_connect_type_e connect_type)

  setter for the 'connect_type' property.
- static GstStateChangeReturn gst_edgesrc_change_state (GstElement *element, GstStateChange transition)

  Change state of edgesrc.
- static void gst_edgesrc_class_init (GstEdgeSrcClass *klass)

  initialize the class
- static void gst_edgesrc_init (GstEdgeSrc *self)

  initialize edgesrc element
- static int _nns_edge_event_cb (nns_edge_event_h event_h, void *user_data)

  nnstreamer-edge event callback.

Variables

- static GstStaticPadTemplate srctemplate

  the capabilities of the outputs

9.12.1 Detailed Description

Subscribe and push incoming data to the GStreamer pipeline.

Date

02 Aug 2022

Author

Yechan Choi yechan9.choi@samsung.com

See also

http://github.com/nnstreamer/nnstreamer

Bug No known bugs

9.12.2 Macro Definition Documentation

9.12.2.1 GST_CAT_DEFAULT

```
#define GST_CAT_DEFAULT gst_edgesrc_debug
```

Definition at line 20 of file edge_src.c.

9.12.2.2 gst_edgesrc_parent_class

```
#define gst_edgesrc_parent_class parent_class
```

Definition at line 45 of file edge_src.c.

9.12.3 Enumeration Type Documentation

9.12.3.1 anonymous enum

```
anonymous enum
```

edgesrc properties

Enumerator

PROP_0	
PROP_HOST	
PROP_PORT	
PROP_DEST_HOST	
PROP_DEST_PORT	
PROP_CONNECT_TYPE	
PROP_TOPIC	
PROP_CUSTOM_LIB	
PROP_LAST	

Definition at line 31 of file edge_src.c.

9.12.4 Function Documentation

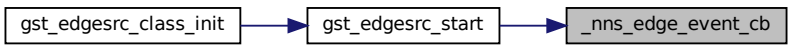
9.12.4.1 _nns_edge_event_cb()

```
static int _nns_edge_event_cb (
            nns_edge_event_h event_h,
            void * user_data )  [static]
```

nnstreamer-edge event callback.

Definition at line 305 of file edge_src.c.

Here is the caller graph for this function:



9.12.4.2 G_DEFINE_TYPE()

```
G_DEFINE_TYPE (
            GstEdgeSrc ,
            gst_edgesrc ,
            GST_TYPE_BASE_SRC  )
```

9.12.4.3 GST_DEBUG_CATEGORY_STATIC()

```
GST_DEBUG_CATEGORY_STATIC (
            gst_edgesrc_debug  )
```

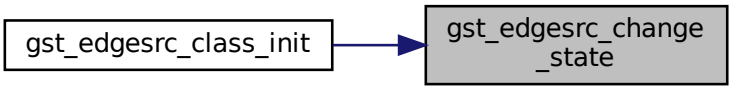
9.12.4.4 gst_edgesrc_change_state()

```
static GstStateChangeReturn gst_edgesrc_change_state (
            GstElement * element,
            GstStateChange transition )  [static]
```

Change state of edgesrc.

Definition at line 273 of file edge_src.c.

Here is the caller graph for this function:



9.12.4.5 gst_edgesrc_class_finalize()

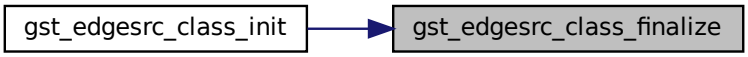
```
static void gst_edgesrc_class_finalize (
            GObject * object )  [static]
```

finalize the object

Definition at line 239 of file edge_src.c.

Here is the call graph for this function:

Here is the caller graph for this function:



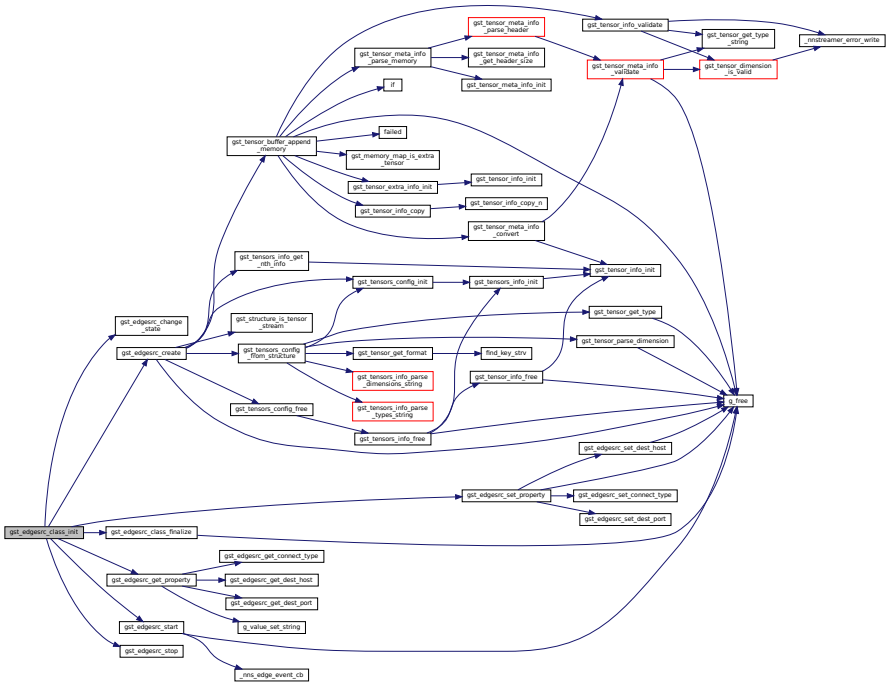
9.12.4.6 gst_edgesrc_class_init()

```
static void gst_edgesrc_class_init (
            GstEdgeSrcClass * klass )  [static]
```

initialize the class

Definition at line 77 of file edge_src.c.

Here is the call graph for this function:



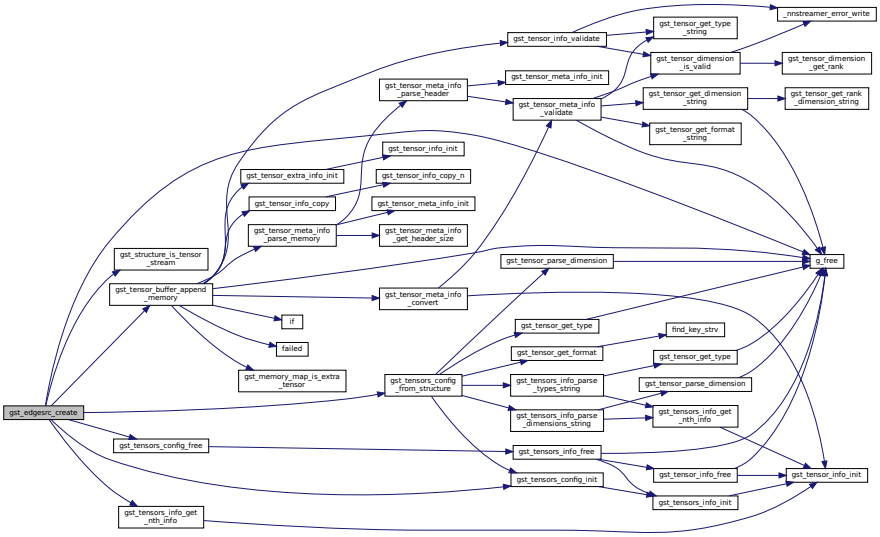
9.12.4.7 gst_edgesrc_create()

```
static GstFlowReturn gst_edgesrc_create (
            GstBaseSrc * basesrc,
            guint64 offset,
            guint size,
            GstBuffer ** out_buf )  [static]
```

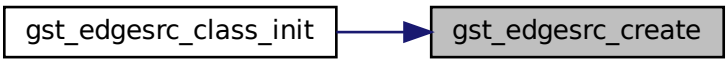
Create a buffer containing the subscribed data.

Definition at line 423 of file edge_src.c.

Here is the call graph for this function:



Here is the caller graph for this function:



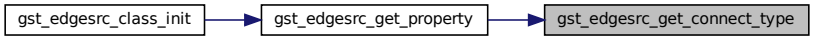
9.12.4.8 gst_edgesrc_get_connect_type()

```
static nns_edge_connect_type_e gst_edgesrc_get_connect_type (
            GstEdgeSrc * self )  [static]
```

getter for the 'connect_type' property.

Definition at line 553 of file edge_src.c.

Here is the caller graph for this function:



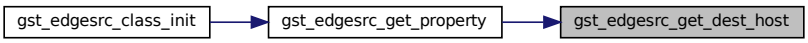
9.12.4.9 gst_edgesrc_get_dest_host()

```
static gchar * gst_edgesrc_get_dest_host (
            GstEdgeSrc * self )  [static]
```

getter for the 'host' property.

Definition at line 516 of file edge_src.c.

Here is the caller graph for this function:



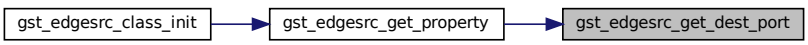
9.12.4.10 gst_edgesrc_get_dest_port()

```
static guint16 gst_edgesrc_get_dest_port (
            GstEdgeSrc * self )  [static]
```

getter for the 'port' property.

Definition at line 535 of file edge_src.c.

Here is the caller graph for this function:



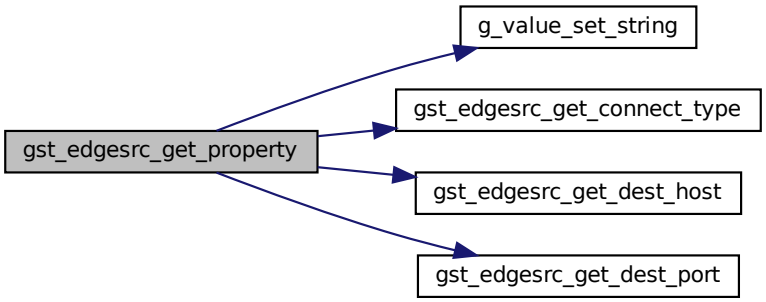
9.12.4.11 gst_edgesrc_get_property()

```
static void gst_edgesrc_get_property (
            GObject * object,
            guint prop_id,
            GValue * value,
            GParamSpec * pspec )  [static]
```

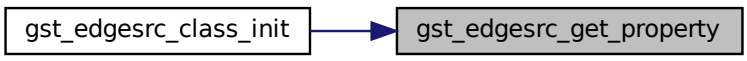
get property

Definition at line 202 of file edge_src.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.12.4.12 gst_edgesrc_init()

```
static void gst_edgesrc_init (
            GstEdgeSrc * self )  [static]
```

initialize edgesrc element

Definition at line 139 of file edge_src.c.

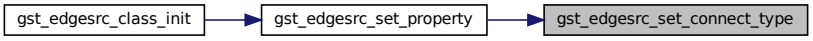
9.12.4.13 gst_edgesrc_set_connect_type()

```
static void gst_edgesrc_set_connect_type (
            GstEdgeSrc * self,
            const nns_edge_connect_type_e connect_type )  [static]
```

setter for the 'connect_type' property.

Definition at line 562 of file edge_src.c.

Here is the caller graph for this function:



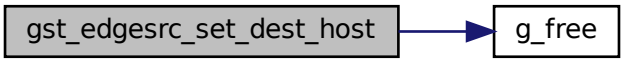
9.12.4.14 gst_edgesrc_set_dest_host()

```
static void gst_edgesrc_set_dest_host (
            GstEdgeSrc * self,
            const gchar * dest_host )  [static]
```

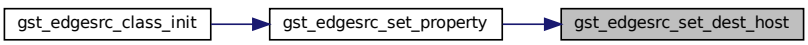
setter for the 'host' property.

Definition at line 525 of file edge_src.c.

Here is the call graph for this function:



Here is the caller graph for this function:



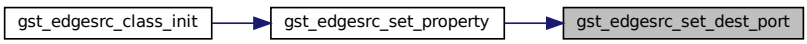
9.12.4.15 gst_edgesrc_set_dest_port()

```
static void gst_edgesrc_set_dest_port (
            GstEdgeSrc * self,
            const guint16 dest_port )  [static]
```

setter for the 'port' property.

Definition at line 544 of file edge_src.c.

Here is the caller graph for this function:



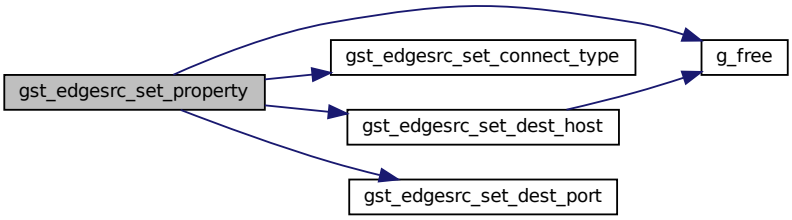
9.12.4.16 gst_edgesrc_set_property()

```
static void gst_edgesrc_set_property (
            GObject * object,
            guint prop_id,
            const GValue * value,
            GParamSpec * pspec )  [static]
```

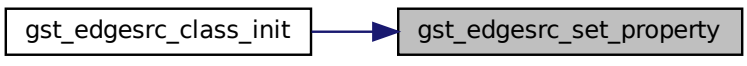
set property

Definition at line 159 of file edge_src.c.

Here is the call graph for this function:



Here is the caller graph for this function:



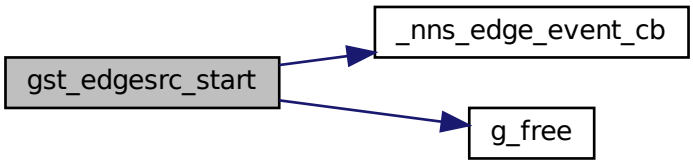
9.12.4.17 gst_edgesrc_start()

```
static gboolean gst_edgesrc_start (
            GstBaseSrc * basesrc )  [static]
```

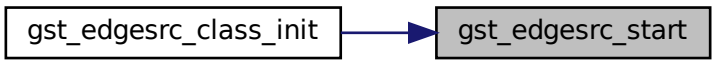
start edgesrc, called when state changed null to ready

Definition at line 342 of file edge_src.c.

Here is the call graph for this function:



Here is the caller graph for this function:



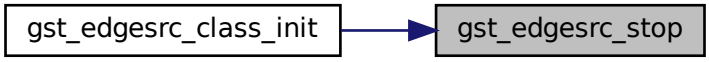
9.12.4.18 gst_edgesrc_stop()

```
static gboolean gst_edgesrc_stop (
            GstBaseSrc * basesrc )  [static]
```

Stop edgesrc, called when state changed ready to null.

Definition at line 403 of file edge_src.c.

Here is the caller graph for this function:



9.12.5 Variable Documentation

9.12.5.1 srctemplate

```
GstStaticPadTemplate srctemplate  [static]
```

Initial value:

```
= GST_STATIC_PAD_TEMPLATE ("src",
    GST_PAD_SRC, GST_PAD_ALWAYS, GST_STATIC_CAPS_ANY)
```

the capabilities of the outputs

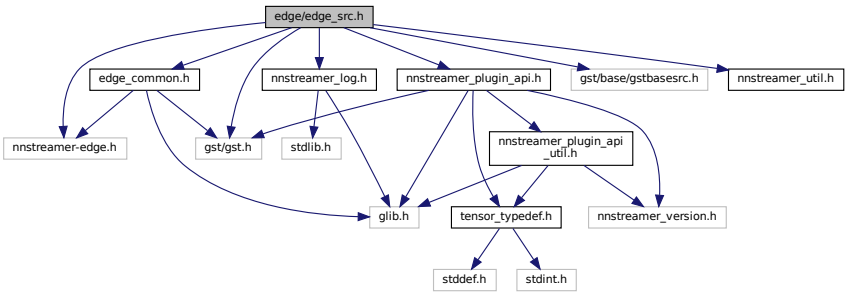
Definition at line 25 of file edge_src.c.

9.13 edge/edge_src.h File Reference

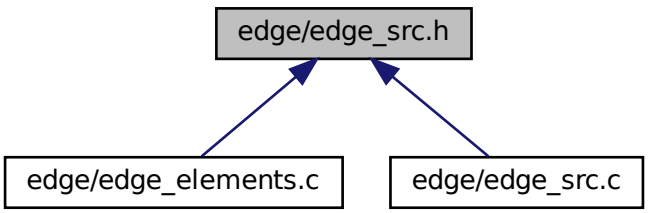
Subscribe and push incoming data to the GStreamer pipeline.

```
#include <gst/gst.h>
#include <gst/base/gstbasesrc.h>
#include "edge_common.h"
#include <nnstreamer-edge.h>
#include "nnstreamer_log.h"
#include "nnstreamer_plugin_api.h"
#include "nnstreamer_util.h"
```

Include dependency graph for edge_src.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct _GstEdgeSrc

  GstEdgeSrc data structure.
- struct _GstEdgeSrcClass

  GstEdgeSrcClass data structure.

Macros

- #define GST_TYPE_EDGESRC (gst_edgesrc_get_type())
- #define GST_EDGESRC(obj) (G_TYPE_CHECK_INSTANCE_CAST((obj),GST_TYPE_EDGESRC,GstEdgeSrc))
- #define GST_EDGESRC_CLASS(klass) (G_TYPE_CHECK_CLASS_CAST((klass),GST_TYPE_EDGESRC,GstEdgeSrcClass
- #define GST_IS_EDGESRC(obj) (G_TYPE_CHECK_INSTANCE_TYPE((obj),GST_TYPE_EDGESRC))
- #define GST_IS_EDGESRC_CLASS(klass) (G_TYPE_CHECK_CLASS_TYPE((klass), GST_TYPE_EDGESRC))
- #define GST_EDGESRC_CAST(obj) ((GstEdgeSrc *) (obj))

Typedefs

- typedef struct _GstEdgeSrc GstEdgeSrc
- typedef struct _GstEdgeSrcClass GstEdgeSrcClass

Functions

- GType gst_edgesrc_get_type (void)

9.13.1 Detailed Description

Subscribe and push incoming data to the GStreamer pipeline.

Date

02 Aug 2022

Author

Yechan Choi yechan9.choi@samsung.com

See also

http://github.com/nnstreamer/nnstreamer

Bug No known bugs

9.13.2 Macro Definition Documentation

9.13.2.1 GST_EDGESRC

```
#define GST_EDGESRC(
              obj ) (G_TYPE_CHECK_INSTANCE_CAST((obj),GST_TYPE_EDGESRC,GstEdgeSrc))
```

Definition at line 27 of file edge_src.h.

9.13.2.2 GST_EDGESRC_CAST

```
#define GST_EDGESRC_CAST(
              obj ) ((GstEdgeSrc *) (obj))
```

Definition at line 35 of file edge_src.h.

9.13.2.3 GST_EDGESRC_CLASS

```
#define GST_EDGESRC_CLASS(
              klass ) (G_TYPE_CHECK_CLASS_CAST((klass),GST_TYPE_EDGESRC,GstEdgeSrcClass))
```

Definition at line 29 of file edge_src.h.

9.13.2.4 GST_IS_EDGESRC

```
#define GST_IS_EDGESRC(
              obj ) (G_TYPE_CHECK_INSTANCE_TYPE((obj),GST_TYPE_EDGESRC))
```

Definition at line 31 of file edge_src.h.

9.13.2.5 GST_IS_EDGESRC_CLASS

```
#define GST_IS_EDGESRC_CLASS(
              klass ) (G_TYPE_CHECK_CLASS_TYPE((klass), GST_TYPE_EDGESRC))
```

Definition at line 33 of file edge_src.h.

9.13.2.6 GST_TYPE_EDGESRC

```
#define GST_TYPE_EDGESRC (gst_edgesrc_get_type())
```

Definition at line 25 of file edge_src.h.

9.13.3 Typedef Documentation

9.13.3.1 GstEdgeSrc

```
typedef struct _GstEdgeSrc GstEdgeSrc
```

Definition at line 36 of file edge_src.h.

9.13.3.2 GstEdgeSrcClass

```
typedef struct _GstEdgeSrcClass GstEdgeSrcClass
```

Definition at line 37 of file edge_src.h.

9.13.4 Function Documentation

9.13.4.1 gst_edgesrc_get_type()

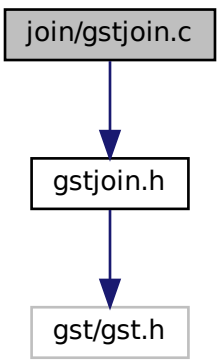
```
GType gst_edgesrc_get_type (
            void  )
```

9.14 join/gstjoin.c File Reference

Select the out that arrived first among the input streams.

```
#include "gstjoin.h"
```

Include dependency graph for gstjoin.c:



Classes

- struct _GstJoinPad

  GstJoinPad data structure.
- struct _GstJoinPadClass

  _GstJoinPadClass data structure

Macros

- #define GST_CAT_DEFAULT join_debug
- #define GST_JOIN_GET_LOCK(sel) (&((GstJoin*)(sel))->lock)
- #define GST_JOIN_GET_COND(sel) (&((GstJoin*)(sel))->cond)
- #define GST_JOIN_LOCK(sel) (g_mutex_lock (GST_JOIN_GET_LOCK(sel)))
- #define GST_JOIN_UNLOCK(sel) (g_mutex_unlock (GST_JOIN_GET_LOCK(sel)))
- #define GST_JOIN_WAIT(sel)
- #define GST_TYPE_JOIN_PAD (gst_join_pad_get_type())
- #define GST_JOIN_PAD(obj) (G_TYPE_CHECK_INSTANCE_CAST ((obj), GST_TYPE_JOIN_PAD, GstJoinPad))
- #define GST_JOIN_PAD_CLASS(klass) (G_TYPE_CHECK_CLASS_CAST ((klass), GST_TYPE_JOIN_PAD, GstJoinPadClass))
- #define GST_IS_JOIN_PAD(obj) (G_TYPE_CHECK_INSTANCE_TYPE ((obj), GST_TYPE_JOIN_PAD))
- #define GST_IS_JOIN_PAD_CLASS(klass) (G_TYPE_CHECK_CLASS_TYPE ((klass), GST_TYPE_JOIN_PAD))
- #define GST_JOIN_PAD_CAST(obj) ((GstJoinPad *)(obj))
- #define gst_join_parent_class parent_class
- #define PACKAGE "join"

Typedefs

- typedef struct _GstJoinPad GstJoinPad
- typedef struct _GstJoinPadClass GstJoinPadClass

Enumerations

- enum { PROP_0, PROP_N_PADS, PROP_ACTIVE_PAD }

Functions

- GST_DEBUG_CATEGORY_STATIC (join_debug)
- static GstPad * gst_join_get_active_sinkpad (GstJoin *sel)

  Get or create the active sinkpad.
- static GstPad * gst_join_get_linked_pad (GstJoin *sel, GstPad *pad, gboolean strict)

  Get linked pad.
- static gboolean gst_join_set_active_pad (GstJoin *self, GstPad *pad)

  set active sink pad.
- GType gst_join_pad_get_type (void)
- static void gst_join_pad_finalize (GObject *object)

  finalize the join pad
- static void gst_join_pad_reset (GstJoinPad *pad)

  Clear and reset join pad.
- static gboolean gst_join_pad_event (GstPad *pad, GstObject *parent, GstEvent *event)

event function for sink pad

- static gboolean gst_join_pad_query (GstPad *pad, GstObject *parent, GstQuery *query)
  handlesink sink pad query
- static GstIterator * gst_join_pad_iterate_linked_pads (GstPad *pad, GstObject *parent)
  strictly get the linked pad from the sinkpad.
- static GstFlowReturn gst_join_pad_chain (GstPad *pad, GstObject *parent, GstBuffer *buf)
  Chain function, this function does the actual processing.
- G_DEFINE_TYPE (GstJoinPad, gst_join_pad, GST_TYPE_PAD)
- static void gst_join_pad_class_init (GstJoinPadClass *klass)
  initialize the join's pad class
- static void gst_join_pad_init (GstJoinPad *pad)
  initialize the join pad
- static gboolean forward_sticky_events (GstPad *sinkpad, GstEvent **event, gpointer user_data)
  forward sticky event
- static void gst_join_dispose (GObject *object)
  dispose function for join element
- static void gst_join_finalize (GObject *object)
  finalize join element.
- static void gst_join_get_property (GObject *object, guint prop_id, GValue *value, GParamSpec *pspec)
  Getter for join properties.
- static GstPad * gst_join_request_new_pad (GstElement *element, GstPadTemplate *templ, const gchar *unused, const GstCaps *caps)
  request new sink pad
- G_DEFINE_TYPE_WITH_CODE (GstJoin, gst_join, GST_TYPE_ELEMENT, GST_DEBUG_CATEGORY↩ _INIT(join_debug, "join", 0, "An input stream join element"))
- static void gst_join_class_init (GstJoinClass *klass)
  initialize the join's class
- static void gst_join_init (GstJoin *sel)
  initialize the join element
- static gboolean plugin_init (GstPlugin *plugin)
  register this element

Variables

- static GstStaticPadTemplate gst_join_sink_factory
  The capabilities of the inputs.
- static GstStaticPadTemplate gst_join_src_factory
  The capabilities of the outputs.

9.14.1 Detailed Description

Select the out that arrived first among the input streams.

Copyright (C) 2020 Gichan Jang gichan2.jang@samsung.com

Date

10 Nov 2020

See also

https://github.com/nnstreamer/nnstreamer

Author

Gichan Jang gichan2.jang@samsung.com

Bug No known bugs except for NYI items

9.14.2 Macro Definition Documentation

9.14.2.1 GST_CAT_DEFAULT

```
#define GST_CAT_DEFAULT join_debug
```

Definition at line 35 of file gstjoin.c.

9.14.2.2 GST_IS_JOIN_PAD

```
#define GST_IS_JOIN_PAD(
              obj ) (G_TYPE_CHECK_INSTANCE_TYPE ((obj), GST_TYPE_JOIN_PAD))
```

Definition at line 80 of file gstjoin.c.

9.14.2.3 GST_IS_JOIN_PAD_CLASS

```
#define GST_IS_JOIN_PAD_CLASS(
              klass ) (G_TYPE_CHECK_CLASS_TYPE ((klass), GST_TYPE_JOIN_PAD))
```

Definition at line 82 of file gstjoin.c.

9.14.2.4 GST_JOIN_GET_COND

```
#define GST_JOIN_GET_COND(
              sel ) (&((GstJoin*)(sel))->cond)
```

Definition at line 38 of file gstjoin.c.

9.14.2.5 GST_JOIN_GET_LOCK

```
#define GST_JOIN_GET_LOCK(
              sel ) (&((GstJoin*)(sel))->lock)
```

Definition at line 37 of file gstjoin.c.

9.14.2.6 GST_JOIN_LOCK

```
#define GST_JOIN_LOCK(
              sel ) (g_mutex_lock (GST_JOIN_GET_LOCK(sel)))
```

Definition at line 39 of file gstjoin.c.

9.14.2.7 GST_JOIN_PAD

```
#define GST_JOIN_PAD(
              obj ) (G_TYPE_CHECK_INSTANCE_CAST ((obj), GST_TYPE_JOIN_PAD, GstJoinPad))
```

Definition at line 76 of file gstjoin.c.

9.14.2.8 GST_JOIN_PAD_CAST

```
#define GST_JOIN_PAD_CAST(
              obj ) ((GstJoinPad *)(obj))
```

Definition at line 84 of file gstjoin.c.

9.14.2.9 GST_JOIN_PAD_CLASS

```
#define GST_JOIN_PAD_CLASS(
              klass ) (G_TYPE_CHECK_CLASS_CAST ((klass), GST_TYPE_JOIN_PAD, GstJoinPadClass))
```

Definition at line 78 of file gstjoin.c.

9.14.2.10 gst_join_parent_class

```
#define gst_join_parent_class parent_class
```

Definition at line 428 of file gstjoin.c.

9.14.2.11 GST_JOIN_UNLOCK

```
#define GST_JOIN_UNLOCK(
              sel ) (g_mutex_unlock (GST_JOIN_GET_LOCK(sel)))
```

Definition at line 40 of file gstjoin.c.

9.14.2.12 GST_JOIN_WAIT

```
#define GST_JOIN_WAIT(
              sel )
```

Value:

```
(g_cond_wait (GST_JOIN_GET_COND(sel), \
GST_JOIN_GET_LOCK(sel)))
```

Definition at line 41 of file gstjoin.c.

9.14.2.13 GST_TYPE_JOIN_PAD

```
#define GST_TYPE_JOIN_PAD (gst_join_pad_get_type())
```

Definition at line 74 of file gstjoin.c.

9.14.2.14 PACKAGE

```
#define PACKAGE "join"
```

Definition at line 701 of file gstjoin.c.

9.14.3 Typedef Documentation

9.14.3.1 GstJoinPad

```
typedef struct _GstJoinPad GstJoinPad
```

Definition at line 87 of file gstjoin.c.

9.14.3.2 GstJoinPadClass

```
typedef struct _GstJoinPadClass GstJoinPadClass
```

Definition at line 88 of file gstjoin.c.

9.14.4 Enumeration Type Documentation

9.14.4.1 anonymous enum

```
anonymous enum
```

Enumerator

PROP_0	
PROP_N_PADS	
PROP_ACTIVE_PAD	

Definition at line 62 of file gstjoin.c.

9.14.5 Function Documentation

9.14.5.1 forward_sticky_events()

```
static gboolean forward_sticky_events (
            GstPad * sinkpad,
            GstEvent ** event,
            gpointer user_data )  [static]
```

forward sticky event

Definition at line 198 of file gstjoin.c.

Here is the caller graph for this function:

9.14.5.2 G_DEFINE_TYPE()

```
G_DEFINE_TYPE (
            GstJoinPad ,
            gst_join_pad ,
            GST_TYPE_PAD  )
```

9.14.5.3 G_DEFINE_TYPE_WITH_CODE()

```
G_DEFINE_TYPE_WITH_CODE (
            GstJoin ,
            gst_join ,
            GST_TYPE_ELEMENT ,
            GST_DEBUG_CATEGORY_INIT(join_debug, "join", 0, "An input stream join element")  )
```

9.14.5.4 GST_DEBUG_CATEGORY_STATIC()

```
GST_DEBUG_CATEGORY_STATIC (
             join_debug  )
```

SECTION:element-join @see_also #GstInputSelector

Note

A join has reduced and changed input-selector's function.

Connect recently arrived buffer from N input streams to the output pad. N streams should not operate at the same time. All capabilities (input stream i and output stream) should be the same. For example, If one sinkpad is receiving buffer, the others should be stopped. <refsect2> <title>Example launch line</title> gst-launch-1.0 ... (input stream 0) ! join.sink_0 \ ... (input stream 1) ! join.sink_1 \ ... \ ... (input stream N) ! join.sink_n \ join name=join ! (arrived input stream) ... </refsect2>

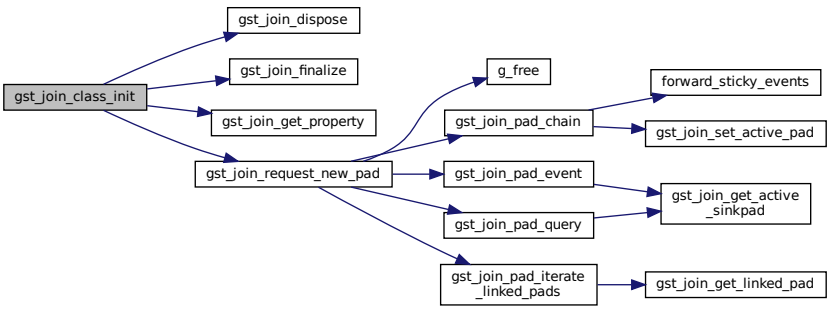
9.14.5.5 gst_join_class_init()

```
static void gst_join_class_init (
             GstJoinClass * klass )  [static]
```

initialize the join's class

Definition at line 437 of file gstjoin.c.

Here is the call graph for this function:



9.14.5.6 gst_join_dispose()

```
static void gst_join_dispose (
            GObject * object )  [static]
```

dispose function for join element

Definition at line 494 of file gstjoin.c.

Here is the caller graph for this function:

9.14.5.7 gst_join_finalize()

```
static void gst_join_finalize (
            GObject * object )  [static]
```

finalize join element.

Definition at line 509 of file gstjoin.c.

Here is the caller graph for this function:

9.14.5.8 gst_join_get_active_sinkpad()

```
static GstPad * gst_join_get_active_sinkpad (
            GstJoin * sel )  [static]
```

Get or create the active sinkpad.

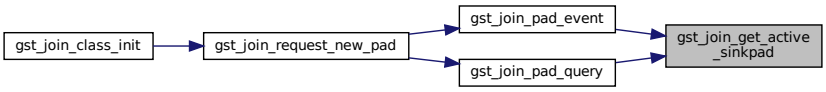
Note

must be called with JOIN_LOCK.

If no pad is currently selected, we return the first usable pad to guarantee consistency

Definition at line 612 of file gstjoin.c.

Here is the caller graph for this function:



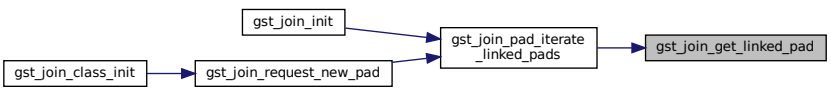
9.14.5.9 gst_join_get_linked_pad()

```
static GstPad * gst_join_get_linked_pad (
            GstJoin * sel,
            GstPad * pad,
            gboolean strict )  [static]
```

Get linked pad.

Definition at line 591 of file gstjoin.c.

Here is the caller graph for this function:



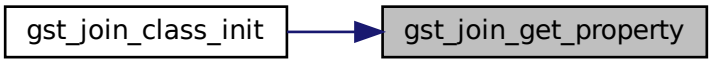
9.14.5.10 gst_join_get_property()

```
static void gst_join_get_property (
             GObject * object,
             guint prop_id,
             GValue * value,
             GParamSpec * pspec )  [static]
```

Getter for join properties.

Definition at line 565 of file gstjoin.c.

Here is the caller graph for this function:



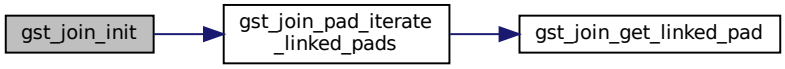
9.14.5.11 gst_join_init()

```
static void gst_join_init (
             GstJoin * sel )  [static]
```

initialize the join element

Definition at line 474 of file gstjoin.c.

Here is the call graph for this function:



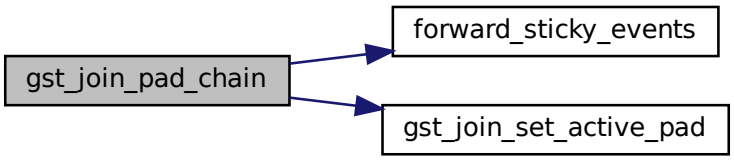
9.14.5.12 gst_join_pad_chain()

```
static GstFlowReturn gst_join_pad_chain (
            GstPad * pad,
            GstObject * parent,
            GstBuffer * buf )  [static]
```

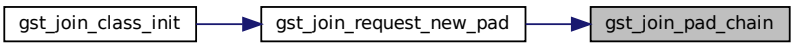
Chain function, this function does the actual processing.

Definition at line 357 of file gstjoin.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.14.5.13 gst_join_pad_class_init()

```
static void gst_join_pad_class_init (
            GstJoinPadClass * klass )  [static]
```

initialize the join's pad class

Definition at line 129 of file gstjoin.c.

Here is the call graph for this function:

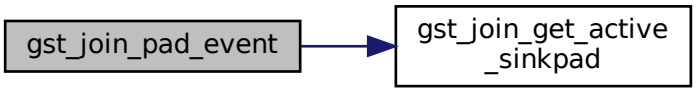
9.14.5.14 gst_join_pad_event()

```
static gboolean gst_join_pad_event (
            GstPad * pad,
            GstObject * parent,
            GstEvent * event )  [static]
```

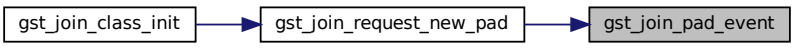
event function for sink pad

Definition at line 233 of file gstjoin.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.14.5.15 gst_join_pad_finalize()

```
static void gst_join_pad_finalize (
            GObject * object )  [static]
```

finalize the join pad

Definition at line 151 of file gstjoin.c.

Here is the caller graph for this function:

9.14.5.16 gst_join_pad_get_type()

```
GType gst_join_pad_get_type (
            void  )
```

9.14.5.17 gst_join_pad_init()

```
static void gst_join_pad_init (
            GstJoinPad * pad )  [static]
```

initialize the join pad

Definition at line 142 of file gstjoin.c.

Here is the call graph for this function:

9.14.5.18 gst_join_pad_iterate_linked_pads()

```
static GstIterator * gst_join_pad_iterate_linked_pads (
            GstPad * pad,
            GstObject * parent )  [static]
```

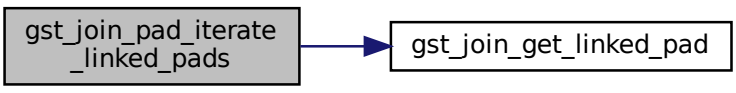
strictly get the linked pad from the sinkpad.

Returns

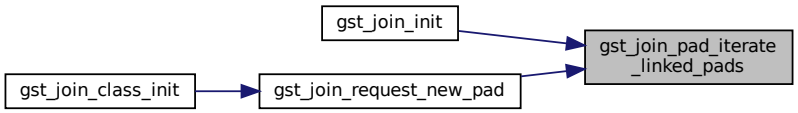
If the pad is active, return the srcpad else return NULL.

Definition at line 173 of file gstjoin.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.14.5.19 gst_join_pad_query()

```
static gboolean gst_join_pad_query (
            GstPad * pad,
            GstObject * parent,
            GstQuery * query )  [static]
```

handlesink sink pad query

Returns

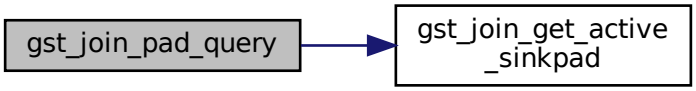
TRUE if the query was performed successfully.

always proxy caps/position/duration/context queries, regardless of active pad or not See https://bugzilla.gnome.org/show_bug.cgi?id=775445

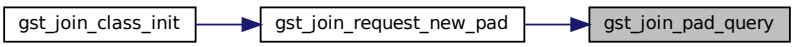
Only do the allocation query for the active sinkpad, after switching a reconfigure event is sent and upstream should reconfigure and do a new allocation query

Definition at line 307 of file gstjoin.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.14.5.20 gst_join_pad_reset()

```
static void gst_join_pad_reset (
            GstJoinPad * pad )  [static]
```

Clear and reset join pad.

Note

must be called with the JOIN_LOCK

Definition at line 161 of file gstjoin.c.

Here is the caller graph for this function:

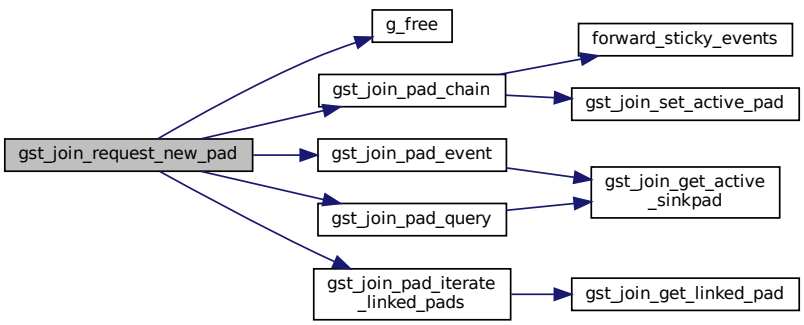
9.14.5.21 gst_join_request_new_pad()

```
static GstPad * gst_join_request_new_pad (
            GstElement * element,
            GstPadTemplate * templ,
            const gchar * unused,
            const GstCaps * caps )  [static]
```

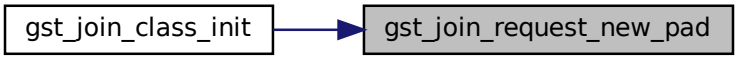
request new sink pad

Definition at line 646 of file gstjoin.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.14.5.22 gst_join_set_active_pad()

```
static gboolean gst_join_set_active_pad (
            GstJoin * self,
            GstPad * pad )  [static]
```

set active sink pad.

Returns

TRUE when the active pad changed.

Note

this function must be called with the JOIN_LOCK.

Definition at line 525 of file gstjoin.c.

Here is the caller graph for this function:

9.14.5.23 plugin_init()

```
static gboolean plugin_init (
            GstPlugin * plugin )  [static]
```

register this element

Definition at line 688 of file gstjoin.c.

9.14.6 Variable Documentation

9.14.6.1 gst_join_sink_factory

```
GstStaticPadTemplate gst_join_sink_factory  [static]
```

Initial value:

```
=
GST_STATIC_PAD_TEMPLATE ("sink_%u",
    GST_PAD_SINK,
    GST_PAD_REQUEST,
    GST_STATIC_CAPS_ANY)
```

The capabilities of the inputs.

Definition at line 47 of file gstjoin.c.

9.14.6.2 gst_join_src_factory

```
GstStaticPadTemplate gst_join_src_factory  [static]
```

Initial value:

```
=
GST_STATIC_PAD_TEMPLATE ("src",
    GST_PAD_SRC,
    GST_PAD_ALWAYS,
    GST_STATIC_CAPS_ANY)
```

The capabilities of the outputs.

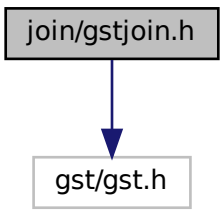
Definition at line 56 of file gstjoin.c.

9.15 join/gstjoin.h File Reference

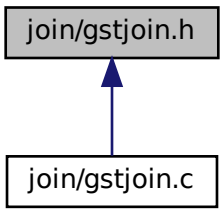
Select the out that arrived first among the input streams.

```
#include <gst/gst.h>
```

Include dependency graph for gstjoin.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct _GstJoin

  Internal data structure for join instances.
- struct _GstJoinClass

  GstJoinClass inherits GstElementClass.

Macros

- #define GST_TYPE_JOIN (gst_join_get_type())
- #define GST_JOIN(obj) (G_TYPE_CHECK_INSTANCE_CAST ((obj), GST_TYPE_JOIN, GstJoin))
- #define GST_JOIN_CLASS(klass) (G_TYPE_CHECK_CLASS_CAST ((klass), GST_TYPE_JOIN, GstJoinClass))
- #define GST_IS_JOIN(obj) (G_TYPE_CHECK_INSTANCE_TYPE ((obj), GST_TYPE_JOIN))
- #define GST_IS_JOIN_CLASS(klass) (G_TYPE_CHECK_CLASS_TYPE ((klass), GST_TYPE_JOIN))

Typedefs

- typedef struct _GstJoin GstJoin
- typedef struct _GstJoinClass GstJoinClass

Functions

- G_GNUC_INTERNAL GType gst_join_get_type (void)

  Get Type function required for gst elements.

9.15.1 Detailed Description

Select the out that arrived first among the input streams.

Date

10 Nov 2020

See also

https://github.com/nnstreamer/nnstreamer

Author

Gichan Jang gichan2.jang@samsung.com

Bug No known bugs except for NYI items

9.15.2 Macro Definition Documentation

9.15.2.1 GST_IS_JOIN

```
#define GST_IS_JOIN(
              obj ) (G_TYPE_CHECK_INSTANCE_TYPE ((obj), GST_TYPE_JOIN))
```

Definition at line 26 of file gstjoin.h.

9.15.2.2 GST_IS_JOIN_CLASS

```
#define GST_IS_JOIN_CLASS(
              klass ) (G_TYPE_CHECK_CLASS_TYPE ((klass), GST_TYPE_JOIN))
```

Definition at line 28 of file gstjoin.h.

9.15.2.3 GST_JOIN

```
#define GST_JOIN(
              obj ) (G_TYPE_CHECK_INSTANCE_CAST ((obj), GST_TYPE_JOIN, GstJoin))
```

Definition at line 22 of file gstjoin.h.

9.15.2.4 GST_JOIN_CLASS

```
#define GST_JOIN_CLASS(
              klass ) (G_TYPE_CHECK_CLASS_CAST ((klass), GST_TYPE_JOIN, GstJoinClass))
```

Definition at line 24 of file gstjoin.h.

9.15.2.5 GST_TYPE_JOIN

```
#define GST_TYPE_JOIN (gst_join_get_type())
```

Definition at line 20 of file gstjoin.h.

9.15.3 Typedef Documentation

9.15.3.1 GstJoin

```
typedef struct _GstJoin GstJoin
```

Definition at line 30 of file gstjoin.h.

9.15.3.2 GstJoinClass

```
typedef struct _GstJoinClass GstJoinClass
```

Definition at line 31 of file gstjoin.h.

9.15.4 Function Documentation

9.15.4.1 gst_join_get_type()

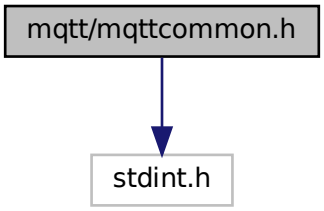
```
G_GNUC_INTERNAL GType gst_join_get_type (
             void  )
```

Get Type function required for gst elements.

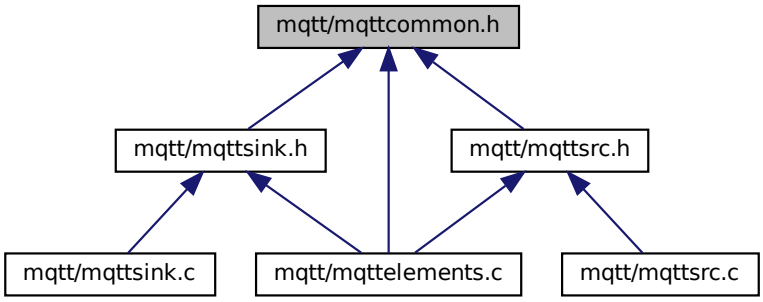
9.16 mqtt/mqttcommon.h File Reference

Common macros and utility functions for GStreamer MQTT plugins.

`#include <stdint.h>`
Include dependency graph for mqttcommon.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct _GstMQTTMessageHdr

  Defined a custom data type, GstMQTTMessageHdr.

Macros

- #define UNUSED(expr) do { (void)(expr); } while (0)
- #define GST_MQTT_PACKAGE "GStreamer MQTT Plugins"
- #define GST_MQTT_ELEM_NAME_SINK "mqttsink"
- #define GST_MQTT_ELEM_NAME_SRC "mqttsrc"
- #define GST_MQTT_LEN_MSG_HDR 1024

- #define GST_MQTT_MAX_LEN_GST_CAPS_STR 512
- #define GST_MQTT_MAX_NUM_MEMS 16

  GST_BUFFER_MEM_MAX in gstreamer/gstbuffer.c is 16. To represent each size of the memory block that the Gst↩ Buffer contains, GST_MQTT_MAX_NUM_MEMS should be 16.
- #define GST_US_TO_NS_MULTIPLIER 1000
- #define DEFAULT_MQTT_CONN_TIMEOUT_SEC 5

Typedefs

- typedef struct _GstMQTTMessageHdr GstMQTTMessageHdr

  Defined a custom data type, GstMQTTMessageHdr.
- typedef int64_t(∗ mqtt_get_unix_epoch) (uint32_t, char ∗∗, uint16_t ∗)

Functions

- static int64_t default_mqtt_get_unix_epoch (uint32_t hnum, char ∗∗hnames, uint16_t ∗hports)

  A wrapper function of g_get_real_time () to assign it to the function pointer, mqtt_get_unix_epoch.

9.16.1 Detailed Description

Common macros and utility functions for GStreamer MQTT plugins.

Copyright (C) 2021 Wook Song wook16.song@samsung.com

Date

08 Mar 2021

See also

https://github.com/nnstreamer/nnstreamer

Author

Wook Song wook16.song@samsung.com

Bug No known bugs except for NYI items

9.16.2 Macro Definition Documentation

9.16.2.1 DEFAULT_MQTT_CONN_TIMEOUT_SEC

```
#define DEFAULT_MQTT_CONN_TIMEOUT_SEC 5
```

Definition at line 40 of file mqttcommon.h.

9.16.2.2 GST_MQTT_ELEM_NAME_SINK

```
#define GST_MQTT_ELEM_NAME_SINK "mqttsink"
```

Definition at line 26 of file mqttcommon.h.

9.16.2.3 GST_MQTT_ELEM_NAME_SRC

```
#define GST_MQTT_ELEM_NAME_SRC "mqttsrc"
```

Definition at line 27 of file mqttcommon.h.

9.16.2.4 GST_MQTT_LEN_MSG_HDR

```
#define GST_MQTT_LEN_MSG_HDR 1024
```

Definition at line 29 of file mqttcommon.h.

9.16.2.5 GST_MQTT_MAX_LEN_GST_CAPS_STR

```
#define GST_MQTT_MAX_LEN_GST_CAPS_STR 512
```

Definition at line 30 of file mqttcommon.h.

9.16.2.6 GST_MQTT_MAX_NUM_MEMS

```
#define GST_MQTT_MAX_NUM_MEMS 16
```

GST_BUFFER_MEM_MAX in gstreamer/gstbuffer.c is 16. To represent each size of the memory block that the GstBuffer contains, GST_MQTT_MAX_NUM_MEMS should be 16.

Definition at line 36 of file mqttcommon.h.

9.16.2.7 GST_MQTT_PACKAGE

```
#define GST_MQTT_PACKAGE "GStreamer MQTT Plugins"
```

Definition at line 23 of file mqttcommon.h.

9.16.2.8 GST_US_TO_NS_MULTIPLIER

```
#define GST_US_TO_NS_MULTIPLIER 1000
```

Definition at line 38 of file mqttcommon.h.

9.16.2.9 UNUSED

```
#define UNUSED(
              expr ) do { (void)(expr); } while (0)
```

Definition at line 19 of file mqttcommon.h.

9.16.3 Typedef Documentation

9.16.3.1 GstMQTTMessageHdr

```
typedef struct _GstMQTTMessageHdr GstMQTTMessageHdr
```

Defined a custom data type, GstMQTTMessageHdr.

GstMQTTMessageHdr contains the information needed to parse the message data at the subscriber side and is prepended to the original message data at the publisher side.

9.16.3.2 mqtt_get_unix_epoch

```
typedef int64_t(* mqtt_get_unix_epoch) (uint32_t, char **, uint16_t *)
```

Definition at line 65 of file mqttcommon.h.

9.16.4 Function Documentation

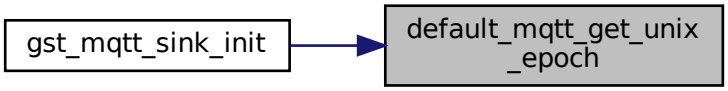
9.16.4.1 default_mqtt_get_unix_epoch()

```
static int64_t default_mqtt_get_unix_epoch (
            uint32_t hnum,
            char ** hnames,
            uint16_t * hports )  [inline], [static]
```

A wrapper function of g_get_real_time () to assign it to the function pointer, mqtt_get_unix_epoch.

Definition at line 71 of file mqttcommon.h.

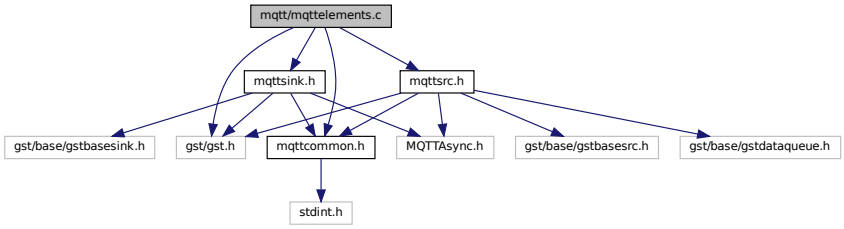
Here is the caller graph for this function:



9.17 mqtt/mqttelements.c File Reference

```
#include <gst/gst.h>
#include "mqttcommon.h"
#include "mqttsink.h"
#include "mqttsrc.h"
```

Include dependency graph for mqttelements.c:



Macros

- #define PACKAGE GST_MQTT_PACKAGE

Functions

- static gboolean plugin_init (GstPlugin *plugin)

  The entry point of the GStreamer MQTT plugin.

9.17.1 Macro Definition Documentation

9.17.1.1 PACKAGE

```
#define PACKAGE GST_MQTT_PACKAGE
```

Definition at line 40 of file mqttelements.c.

9.17.2 Function Documentation

9.17.2.1 plugin_init()

```
static gboolean plugin_init (
            GstPlugin * plugin )  [static]
```

The entry point of the GStreamer MQTT plugin.

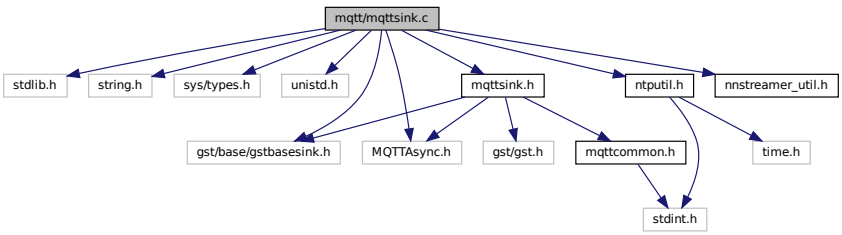
Definition at line 24 of file mqttelements.c.

9.18 mqtt/mqttsink.c File Reference

Register sub-plugins included in libgstmqtt.

```
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <unistd.h>
#include <gst/base/gstbasesink.h>
#include <MQTTAsync.h>
#include <nnstreamer_util.h>
#include "mqttsink.h"
#include "ntputil.h"
```

Include dependency graph for mqttsink.c:



Macros

- #define gst_mqtt_sink_parent_class parent_class
- #define GST_CAT_DEFAULT gst_mqtt_sink_debug

Enumerations

- enum {
  PROP_0, PROP_DEBUG, PROP_MQTT_CLIENT_ID, PROP_MQTT_HOST_ADDRESS,
  PROP_MQTT_HOST_PORT, PROP_MQTT_PUB_TOPIC, PROP_MQTT_PUB_WAIT_TIMEOUT, PROP_MQTT_OPT_CLEA
  PROP_MQTT_OPT_KEEP_ALIVE_INTERVAL, PROP_NUM_BUFFERS, PROP_MAX_MSG_BUF_SIZE,
  PROP_MQTT_QOS,
  PROP_MQTT_NTP_SYNC, PROP_MQTT_NTP_SRVS, PROP_LAST }
- enum {
  DEFAULT_DEBUG = FALSE, DEFAULT_NUM_BUFFERS = -1, DEFAULT_QOS = TRUE, DEFAULT_SYNC
  = FALSE,
  DEFAULT_MQTT_OPT_CLEANSESSION = TRUE, DEFAULT_MQTT_OPT_KEEP_ALIVE_INTERVAL =
  60, DEFAULT_MQTT_DISCONNECT_TIMEOUT = G_TIME_SPAN_SECOND * 3, DEFAULT_MQTT_PUB_WAIT_TIMEOUT
  = 1,
  DEFAULT_MAX_MSG_BUF_SIZE = 0, DEFAULT_MQTT_QOS = 0, DEFAULT_MQTT_NTP_SYNC =
  FALSE, MAX_LEN_PROP_NTP_SRVS = 4096 }

Functions

- G_DEFINE_TYPE (GstMqttSink, gst_mqtt_sink, GST_TYPE_BASE_SINK)
- GST_DEBUG_CATEGORY_STATIC (gst_mqtt_sink_debug)
- static void gst_mqtt_sink_set_property (GObject *object, guint prop_id, const GValue *value, GParamSpec *pspec)

  The setter for the mqttsink's properties.
- static void gst_mqtt_sink_get_property (GObject *object, guint prop_id, GValue *value, GParamSpec *pspec)

  The getter for the mqttsink's properties.
- static void gst_mqtt_sink_class_finalize (GObject *object)

  Finalize GstMqttSinkClass object.
- static GstStateChangeReturn gst_mqtt_sink_change_state (GstElement *element, GstStateChange transition)

  Handle mqttsink's state change.
- static gboolean gst_mqtt_sink_start (GstBaseSink *basesink)

  Start mqttsink, called when state changed null to ready.
- static gboolean gst_mqtt_sink_stop (GstBaseSink *basesink)

  Stop mqttsink, called when state changed ready to null.
- static gboolean gst_mqtt_sink_query (GstBaseSink *basesink, GstQuery *query)

  Perform queries on the element.
- static GstFlowReturn gst_mqtt_sink_render (GstBaseSink *basesink, GstBuffer *in_buf)

  The callback to process each buffer receiving on the sink pad.
- static GstFlowReturn gst_mqtt_sink_render_list (GstBaseSink *basesink, GstBufferList *list)

  The callback to process GstBufferList (instead of a single buffer) on the sink pad.
- static gboolean gst_mqtt_sink_event (GstBaseSink *basesink, GstEvent *event)

  Handle events arriving on the sink pad.
- static gboolean gst_mqtt_sink_set_caps (GstBaseSink *basesink, GstCaps *caps)

  An implementation of the set_caps vmethod in GstBaseSinkClass.
- static gboolean gst_mqtt_sink_get_debug (GstMqttSink *self)

Getter for the 'debug' property.

- static void gst_mqtt_sink_set_debug (GstMqttSink *self, const gboolean flag)

  Setter for the 'debug' property.
- static gchar * gst_mqtt_sink_get_client_id (GstMqttSink *self)

  Getter for the 'client-id' property.
- static void gst_mqtt_sink_set_client_id (GstMqttSink *self, const gchar *id)

  Setter for the 'client-id' property.
- static gchar * gst_mqtt_sink_get_host_address (GstMqttSink *self)

  Getter for the 'host' property.
- static void gst_mqtt_sink_set_host_address (GstMqttSink *self, const gchar *addr)

  Setter for the 'host' property.
- static gchar * gst_mqtt_sink_get_host_port (GstMqttSink *self)

  Getter for the 'port' property.
- static void gst_mqtt_sink_set_host_port (GstMqttSink *self, const gchar *port)

  Setter for the 'port' property.
- static gchar * gst_mqtt_sink_get_pub_topic (GstMqttSink *self)

  Getter for the 'pub-topic' property.
- static void gst_mqtt_sink_set_pub_topic (GstMqttSink *self, const gchar *topic)

  Setter for the 'pub-topic' property.
- static gulong gst_mqtt_sink_get_pub_wait_timeout (GstMqttSink *self)

  Getter for the 'pub-wait-timeout' property.
- static void gst_mqtt_sink_set_pub_wait_timeout (GstMqttSink *self, const gulong to)

  Setter for the 'pub-wait-timeout' property.
- static gboolean gst_mqtt_sink_get_opt_cleansession (GstMqttSink *self)

  Getter for the 'cleansession' property.
- static void gst_mqtt_sink_set_opt_cleansession (GstMqttSink *self, const gboolean val)

  Setter for the 'cleansession' property.
- static gint gst_mqtt_sink_get_opt_keep_alive_interval (GstMqttSink *self)

  Getter for the 'keep-alive-interval' property.
- static void gst_mqtt_sink_set_opt_keep_alive_interval (GstMqttSink *self, const gint num)

  Setter for the 'keep-alive-interval' property.
- static gsize gst_mqtt_sink_get_max_msg_buf_size (GstMqttSink *self)

  Getter for the 'max-buffer-size' property.
- static void gst_mqtt_sink_set_max_msg_buf_size (GstMqttSink *self, const gsize size)

  Setter for the 'max-buffer-size' property.
- static gint gst_mqtt_sink_get_num_buffers (GstMqttSink *self)

  Getter for the 'num-buffers' property.
- static void gst_mqtt_sink_set_num_buffers (GstMqttSink *self, const gint num)

  Setter for the 'num-buffers' property.
- static gint gst_mqtt_sink_get_mqtt_qos (GstMqttSink *self)

  Getter for the 'mqtt-qos' property.
- static void gst_mqtt_sink_set_mqtt_qos (GstMqttSink *self, const gint qos)

  Setter for the 'mqtt-qos' property.
- static gboolean gst_mqtt_sink_get_mqtt_ntp_sync (GstMqttSink *self)

  Getter for the 'ntp-sync' property.
- static void gst_mqtt_sink_set_mqtt_ntp_sync (GstMqttSink *self, const gboolean flag)

  Setter for the 'ntp-sync' property.
- static gchar * gst_mqtt_sink_get_mqtt_ntp_srvs (GstMqttSink *self)

  Getter for the 'ntp-srvs' property.
- static void gst_mqtt_sink_set_mqtt_ntp_srvs (GstMqttSink *self, const gchar *pairs)

  Setter for the 'ntp-srvs' property.

- static void cb_mqtt_on_connect (void *context, MQTTAsync_successData *response)

  A callback function corresponding to MQTTAsync_connectOptions's onSuccess. This callback is invoked when the connection between this element and the broker is properly established.
- static void cb_mqtt_on_connect_failure (void *context, MQTTAsync_failureData *response)

  A callback function corresponding to MQTTAsync_connectOptions's onFailure. This callback is invoked when it is failed to connect to the broker.
- static void cb_mqtt_on_disconnect (void *context, MQTTAsync_successData *response)

  A callback function corresponding to MQTTAsync_disconnectOptions's onSuccess. Regardless of the MQTTAsync↩ _disconnect function's result, the pipeline should be stopped after this callback.
- static void cb_mqtt_on_disconnect_failure (void *context, MQTTAsync_failureData *response)

  A callback function corresponding to MQTTAsync_disconnectOptions's onFailure. Regardless of the MQTTAsync↩ _disconnect function's result, the pipeline should be stopped after this callback.
- static void cb_mqtt_on_delivery_complete (void *context, MQTTAsync_token token)

  A callback function to be given to the MQTTAsync_setCallbacks function. This callback is activated when `mqtt-qos` *is higher then 0.*
- static void cb_mqtt_on_connection_lost (void *context, char *cause)

  A callback function to be given to the MQTTAsync_setCallbacks function. When the connection between this element and the broker is broken, this callback will be invoked.
- static int cb_mqtt_on_message_arrived (void *context, char *topicName, int topicLen, MQTTAsync_message *message)

  A callback function to be given to the MQTTAsync_setCallbacks function. In the case of the publisher, this callback is not used.
- static void cb_mqtt_on_send_success (void *context, MQTTAsync_successData *response)

  A callback function corresponding to MQTTAsync_responseOptions's onSuccess.
- static void cb_mqtt_on_send_failure (void *context, MQTTAsync_failureData *response)

  A callback function corresponding to MQTTAsync_responseOptions's onFailure.
- static void gst_mqtt_sink_init (GstMqttSink *self)

  Initialize GstMqttSink object.
- static void gst_mqtt_sink_class_init (GstMqttSinkClass *klass)

  Initialize GstMqttSinkClass object.
- static void _put_timestamp_to_msg_buf_hdr (GstMqttSink *self, GstBuffer *gst_buf, GstMQTTMessageHdr *hdr)

  A utility function to set the timestamp information onto the given buffer.
- static gboolean _mqtt_set_msg_buf_hdr (GstBuffer *gst_buf, GstMQTTMessageHdr *hdr)

  A utility function to set the message header.

Variables

- static GstStaticPadTemplate sink_pad_template
- static guint8 sink_client_id = 0
- static const gchar DEFAULT_MQTT_HOST_ADDRESS [ ] = "127.0.0.1"
- static const gchar DEFAULT_MQTT_HOST_PORT [ ] = "1883"
- static const gchar TAG_ERR_MQTTSINK [ ] = "ERROR: MQTTSink"
- static const gchar DEFAULT_MQTT_CLIENT_ID [ ] = "$HOST_$PID_^[0-9][0-9]?$|^255$"
- static const gchar DEFAULT_MQTT_CLIENT_ID_FORMAT [ ] = "%s_%u_sink%u"
- static const gchar DEFAULT_MQTT_PUB_TOPIC [ ] = "$client-id/topic"
- static const gchar DEFAULT_MQTT_PUB_TOPIC_FORMAT [ ] = "%s/topic"
- static const gchar DEFAULT_MQTT_NTP_SERVERS [ ] = "pool.ntp.org:123"

9.18.1 Detailed Description

Register sub-plugins included in libgstmqtt.

Publish incoming data streams as a MQTT topic.

Copyright (C) 2021 Wook Song wook16.song@samsung.com

Date

09 Mar 2021

See also

https://github.com/nnstreamer/nnstreamer

Author

Wook Song wook16.song@samsung.com

Bug No known bugs except for NYI items

Copyright (C) 2021 Wook Song wook16.song@samsung.com

Date

01 Apr 2021

See also

https://github.com/nnstreamer/nnstreamer

Author

Wook Song wook16.song@samsung.com

Bug No known bugs except for NYI items

9.18.2 Macro Definition Documentation

9.18.2.1 GST_CAT_DEFAULT

```
#define GST_CAT_DEFAULT gst_mqtt_sink_debug
```

Definition at line 42 of file mqttsink.c.

9.18.2.2 gst_mqtt_sink_parent_class

```
#define gst_mqtt_sink_parent_class parent_class
```

Definition at line 38 of file mqttsink.c.

9.18.3 Enumeration Type Documentation

9.18.3.1 anonymous enum

```
anonymous enum
```

Enumerator

DEFAULT_DEBUG	
DEFAULT_NUM_BUFFERS	
DEFAULT_QOS	
DEFAULT_SYNC	
DEFAULT_MQTT_OPT_CLEANSESSION	
DEFAULT_MQTT_OPT_KEEP_ALIVE_INTERVAL	
DEFAULT_MQTT_DISCONNECT_TIMEOUT	
DEFAULT_MQTT_PUB_WAIT_TIMEOUT	
DEFAULT_MAX_MSG_BUF_SIZE	
DEFAULT_MQTT_QOS	
DEFAULT_MQTT_NTP_SYNC	
MAX_LEN_PROP_NTP_SRVS	

Definition at line 65 of file mqttsink.c.

9.18.3.2 anonymous enum

```
anonymous enum
```

Enumerator

PROP_0	
PROP_DEBUG	
PROP_MQTT_CLIENT_ID	
PROP_MQTT_HOST_ADDRESS	
PROP_MQTT_HOST_PORT	
PROP_MQTT_PUB_TOPIC	
PROP_MQTT_PUB_WAIT_TIMEOUT	
PROP_MQTT_OPT_CLEANSESSION	
PROP_MQTT_OPT_KEEP_ALIVE_INTERVAL	
PROP_NUM_BUFFERS	
PROP_MAX_MSG_BUF_SIZE	
PROP_MQTT_QOS	
PROP_MQTT_NTP_SYNC	
PROP_MQTT_NTP_SRVS	
PROP_LAST	

Definition at line 44 of file mqttsink.c.

9.18.4 Function Documentation

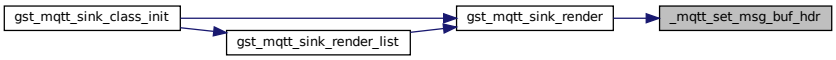
9.18.4.1 _mqtt_set_msg_buf_hdr()

```
static gboolean _mqtt_set_msg_buf_hdr (
            GstBuffer * gst_buf,
            GstMQTTMessageHdr * hdr )  [static]
```

A utility function to set the message header.

Definition at line 725 of file mqttsink.c.

Here is the caller graph for this function:



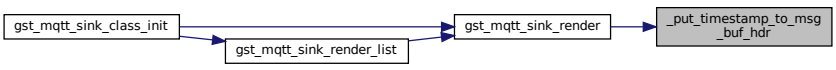
9.18.4.2 _put_timestamp_to_msg_buf_hdr()

```
static void _put_timestamp_to_msg_buf_hdr (
            GstMqttSink * self,
            GstBuffer * gst_buf,
            GstMQTTMessageHdr * hdr )  [static]
```

A utility function to set the timestamp information onto the given buffer.

Definition at line 688 of file mqttsink.c.

Here is the caller graph for this function:



9.18.4.3 cb_mqtt_on_connect()

```
static void cb_mqtt_on_connect (
            void * context,
            MQTTAsync_successData * response )  [static]
```

A callback function corresponding to MQTTAsync_connectOptions's onSuccess. This callback is invoked when the connection between this element and the broker is properly established.

Callback function definitions

Definition at line 1267 of file mqttsink.c.

Here is the caller graph for this function:

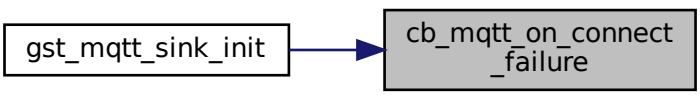
9.18.4.4 cb_mqtt_on_connect_failure()

```
static void cb_mqtt_on_connect_failure (
            void * context,
            MQTTAsync_failureData * response )  [static]
```

A callback function corresponding to MQTTAsync_connectOptions's onFailure. This callback is invoked when it is failed to connect to the broker.

Definition at line 1285 of file mqttsink.c.

Here is the caller graph for this function:



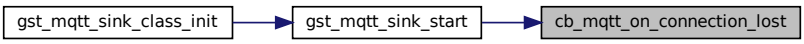
9.18.4.5 cb_mqtt_on_connection_lost()

```
static void cb_mqtt_on_connection_lost (
            void * context,
            char * cause )  [static]
```

A callback function to be given to the MQTTAsync_setCallbacks function. When the connection between this element and the broker is broken, this callback will be invoked.

Definition at line 1351 of file mqttsink.c.

Here is the caller graph for this function:



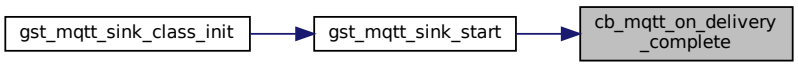
9.18.4.6 cb_mqtt_on_delivery_complete()

```
static void cb_mqtt_on_delivery_complete (
            void * context,
            MQTTAsync_token token )  [static]
```

A callback function to be given to the MQTTAsync_setCallbacks function. This callback is activated when `mqtt-qos` is higher then 0.

Definition at line 1336 of file mqttsink.c.

Here is the caller graph for this function:



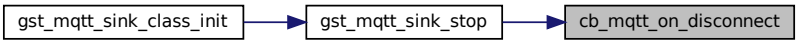
9.18.4.7 cb_mqtt_on_disconnect()

```
static void cb_mqtt_on_disconnect (
            void * context,
            MQTTAsync_successData * response )  [static]
```

A callback function corresponding to MQTTAsync_disconnectOptions's onSuccess. Regardless of the MQTT↩ Async_disconnect function's result, the pipeline should be stopped after this callback.

Definition at line 1303 of file mqttsink.c.

Here is the caller graph for this function:



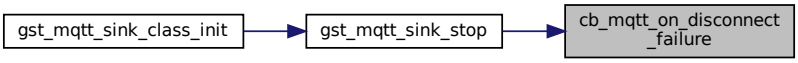
9.18.4.8 cb_mqtt_on_disconnect_failure()

```
static void cb_mqtt_on_disconnect_failure (
            void * context,
            MQTTAsync_failureData * response )  [static]
```

A callback function corresponding to MQTTAsync_disconnectOptions's onFailure. Regardless of the MQTTAsync↩ _disconnect function's result, the pipeline should be stopped after this callback.

Definition at line 1320 of file mqttsink.c.

Here is the caller graph for this function:



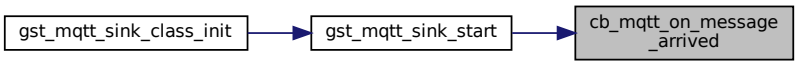
9.18.4.9 cb_mqtt_on_message_arrived()

```
static int cb_mqtt_on_message_arrived (
             void * context,
             char * topicName,
             int topicLen,
             MQTTAsync_message * message )  [static]
```

A callback function to be given to the MQTTAsync_setCallbacks function. In the case of the publisher, this callback is not used.

Definition at line 1368 of file mqttsink.c.

Here is the caller graph for this function:



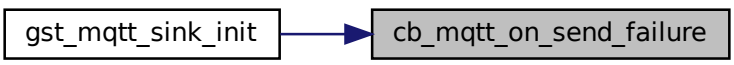
9.18.4.10 cb_mqtt_on_send_failure()

```
static void cb_mqtt_on_send_failure (
             void * context,
             MQTTAsync_failureData * response )  [static]
```

A callback function corresponding to MQTTAsync_responseOptions's onFailure.

Definition at line 1404 of file mqttsink.c.

Here is the caller graph for this function:



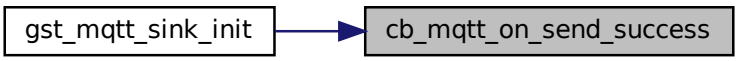
9.18.4.11 cb_mqtt_on_send_success()

```
static void cb_mqtt_on_send_success (
            void * context,
            MQTTAsync_successData * response )  [static]
```

A callback function corresponding to MQTTAsync_responseOptions's onSuccess.

Definition at line 1384 of file mqttsink.c.

Here is the caller graph for this function:



9.18.4.12 G_DEFINE_TYPE()

```
G_DEFINE_TYPE (
            GstMqttSink ,
            gst_mqtt_sink ,
            GST_TYPE_BASE_SINK  )
```

9.18.4.13 GST_DEBUG_CATEGORY_STATIC()

```
GST_DEBUG_CATEGORY_STATIC (
            gst_mqtt_sink_debug  )
```

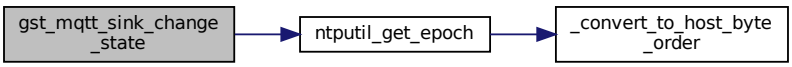
9.18.4.14 gst_mqtt_sink_change_state()

```
static GstStateChangeReturn gst_mqtt_sink_change_state (
            GstElement * element,
            GstStateChange transition )  [static]
```

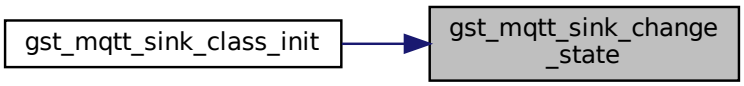
Handle mqttsink's state change.

Definition at line 491 of file mqttsink.c.

Here is the call graph for this function:



Here is the caller graph for this function:



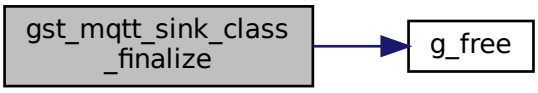
9.18.4.15 gst_mqtt_sink_class_finalize()

```
static void gst_mqtt_sink_class_finalize (
            GObject * object )  [static]
```

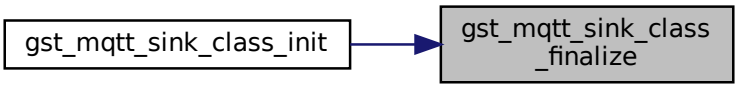
Finalize GstMqttSinkClass object.

Definition at line 454 of file mqttsink.c.

Here is the call graph for this function:



Here is the caller graph for this function:



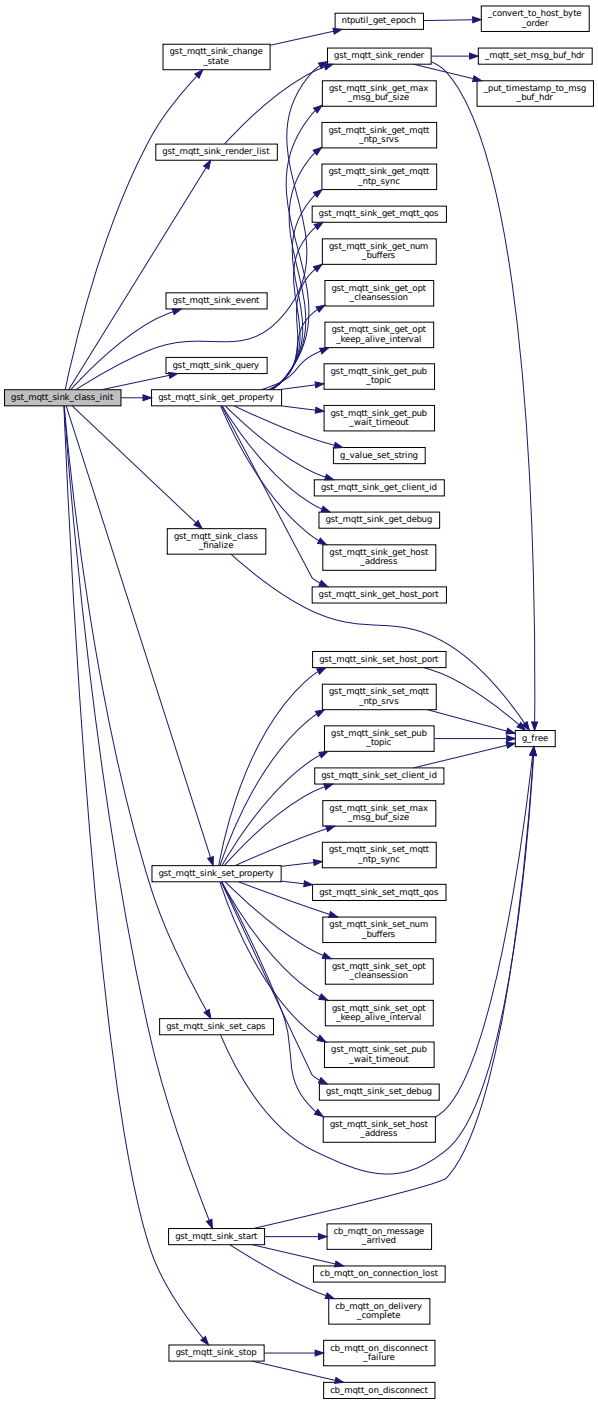
9.18.4.16 gst_mqtt_sink_class_init()

```
static void gst_mqtt_sink_class_init (
            GstMqttSinkClass * klass )  [static]
```

Initialize GstMqttSinkClass object.

Definition at line 229 of file mqttsink.c.

Here is the call graph for this function:



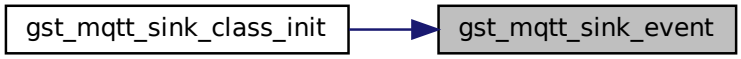
9.18.4.17 gst_mqtt_sink_event()

```
static gboolean gst_mqtt_sink_event (
            GstBaseSink * basesink,
            GstEvent * event )  [static]
```

Handle events arriving on the sink pad.

Definition at line 898 of file mqttsink.c.

Here is the caller graph for this function:



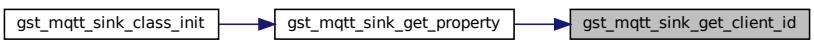
9.18.4.18 gst_mqtt_sink_get_client_id()

```
static gchar * gst_mqtt_sink_get_client_id (
            GstMqttSink * self )  [static]
```

Getter for the 'client-id' property.

Definition at line 976 of file mqttsink.c.

Here is the caller graph for this function:



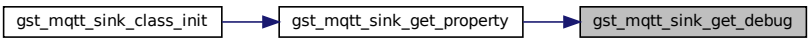
9.18.4.19 gst_mqtt_sink_get_debug()

```
static gboolean gst_mqtt_sink_get_debug (
            GstMqttSink * self )  [static]
```

Getter for the 'debug' property.

Definition at line 958 of file mqttsink.c.

Here is the caller graph for this function:



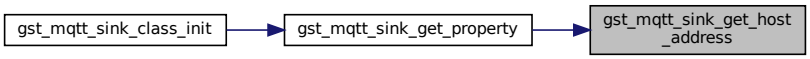
9.18.4.20 gst_mqtt_sink_get_host_address()

```
static gchar * gst_mqtt_sink_get_host_address (
            GstMqttSink * self )  [static]
```

Getter for the 'host' property.

Definition at line 995 of file mqttsink.c.

Here is the caller graph for this function:



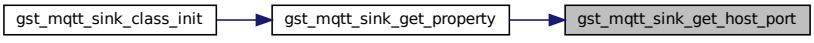
9.18.4.21 gst_mqtt_sink_get_host_port()

```
static gchar * gst_mqtt_sink_get_host_port (
            GstMqttSink * self )  [static]
```

Getter for the 'port' property.

Definition at line 1017 of file mqttsink.c.

Here is the caller graph for this function:



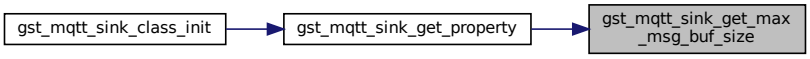
9.18.4.22 gst_mqtt_sink_get_max_msg_buf_size()

```
static gsize gst_mqtt_sink_get_max_msg_buf_size (
            GstMqttSink * self )  [static]
```

Getter for the 'max-buffer-size' property.

Definition at line 1109 of file mqttsink.c.

Here is the caller graph for this function:



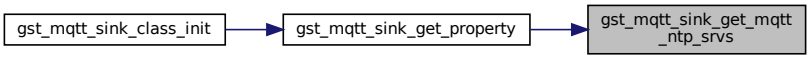
9.18.4.23 gst_mqtt_sink_get_mqtt_ntp_srvs()

```
static gchar * gst_mqtt_sink_get_mqtt_ntp_srvs (
            GstMqttSink * self )  [static]
```

Getter for the 'ntp-srvs' property.

Definition at line 1185 of file mqttsink.c.

Here is the caller graph for this function:



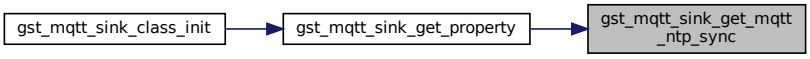
9.18.4.24 gst_mqtt_sink_get_mqtt_ntp_sync()

```
static gboolean gst_mqtt_sink_get_mqtt_ntp_sync (
            GstMqttSink * self )  [static]
```

Getter for the 'ntp-sync' property.

Definition at line 1167 of file mqttsink.c.

Here is the caller graph for this function:



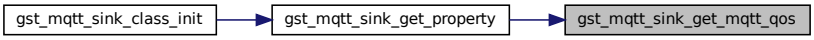
9.18.4.25 gst_mqtt_sink_get_mqtt_qos()

```
static gint gst_mqtt_sink_get_mqtt_qos (
            GstMqttSink * self )  [static]
```

Getter for the 'mqtt-qos' property.

Definition at line 1149 of file mqttsink.c.

Here is the caller graph for this function:



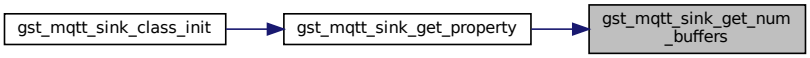
9.18.4.26 gst_mqtt_sink_get_num_buffers()

```
static gint gst_mqtt_sink_get_num_buffers (
            GstMqttSink * self )  [static]
```

Getter for the 'num-buffers' property.

Definition at line 1127 of file mqttsink.c.

Here is the caller graph for this function:



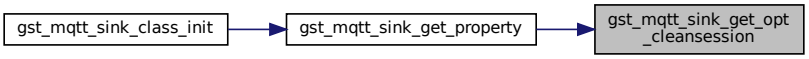
9.18.4.27 gst_mqtt_sink_get_opt_cleansession()

```
static gboolean gst_mqtt_sink_get_opt_cleansession (
            GstMqttSink * self )  [static]
```

Getter for the 'cleansession' property.

Definition at line 1055 of file mqttsink.c.

Here is the caller graph for this function:



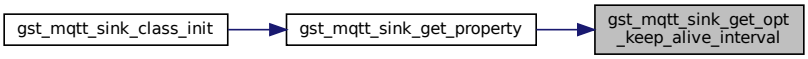
9.18.4.28 gst_mqtt_sink_get_opt_keep_alive_interval()

```
static gint gst_mqtt_sink_get_opt_keep_alive_interval (
            GstMqttSink * self )  [static]
```

Getter for the 'keep-alive-interval' property.

Definition at line 1091 of file mqttsink.c.

Here is the caller graph for this function:



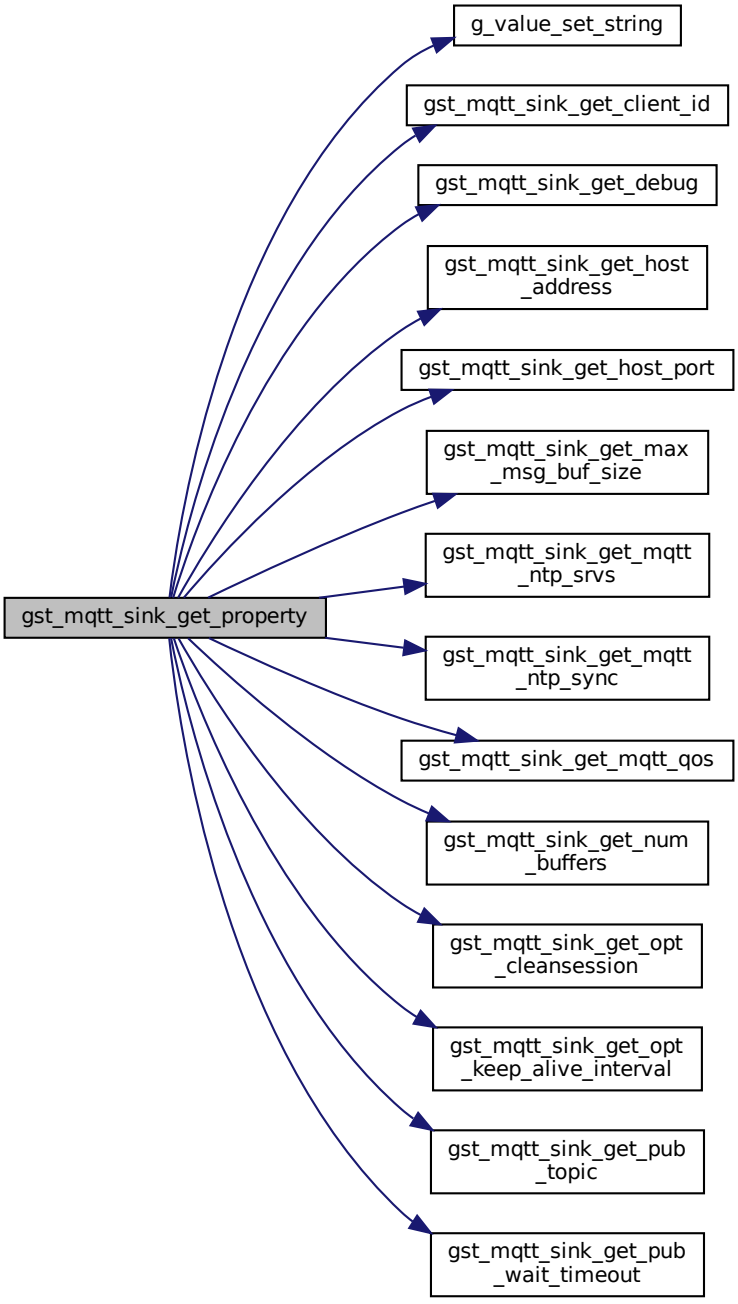
9.18.4.29 gst_mqtt_sink_get_property()

```
static void gst_mqtt_sink_get_property (
            GObject * object,
            guint prop_id,
            GValue * value,
            GParamSpec * pspec )  [static]
```

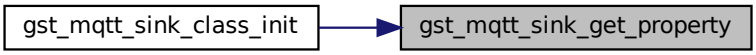
The getter for the mqttsink's properties.

Definition at line 399 of file mqttsink.c.

Here is the call graph for this function:



Here is the caller graph for this function:



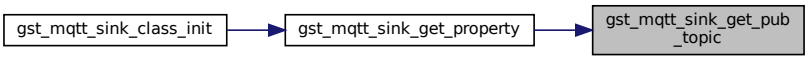
9.18.4.30 gst_mqtt_sink_get_pub_topic()

```
static gchar * gst_mqtt_sink_get_pub_topic (
            GstMqttSink * self )  [static]
```

Getter for the 'pub-topic' property.

Definition at line 1036 of file mqttsink.c.

Here is the caller graph for this function:



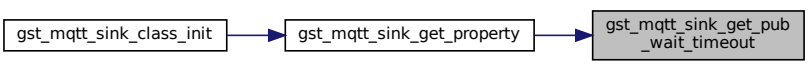
9.18.4.31 gst_mqtt_sink_get_pub_wait_timeout()

```
static gulong gst_mqtt_sink_get_pub_wait_timeout (
            GstMqttSink * self )  [static]
```

Getter for the 'pub-wait-timeout' property.

Definition at line 1073 of file mqttsink.c.

Here is the caller graph for this function:



9.18.4.32 gst_mqtt_sink_init()

```
static void gst_mqtt_sink_init (
            GstMqttSink * self )  [static]
```

Initialize GstMqttSink object.

init MQTT related variables

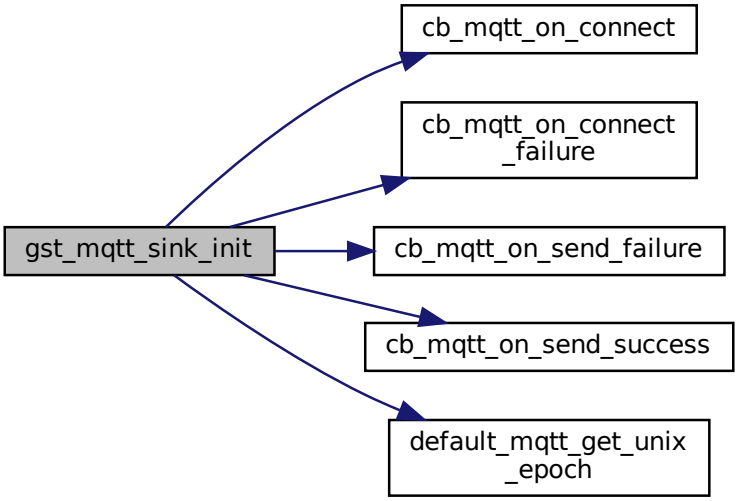
init private variables

init mqttsink properties

init basesink properties

Definition at line 171 of file mqttsink.c.

Here is the call graph for this function:



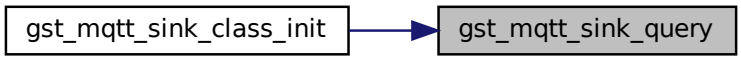
9.18.4.33 gst_mqtt_sink_query()

```
static gboolean gst_mqtt_sink_query (
            GstBaseSink * basesink,
            GstQuery * query )  [static]
```

Perform queries on the element.

Definition at line 661 of file mqttsink.c.

Here is the caller graph for this function:



9.18.4.34 gst_mqtt_sink_render()

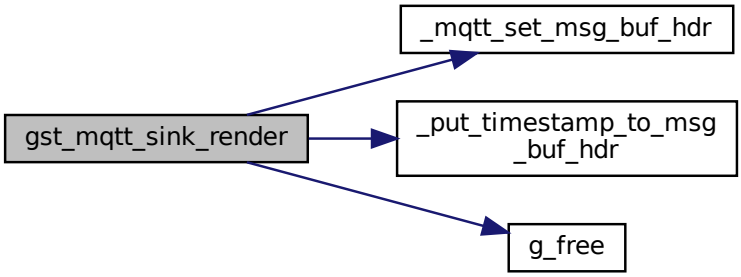
```
static GstFlowReturn gst_mqtt_sink_render (
            GstBaseSink * basesink,
            GstBuffer * buffer )  [static]
```

The callback to process each buffer receiving on the sink pad.

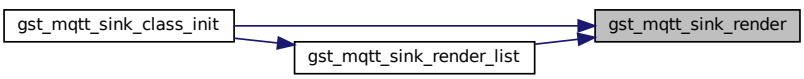
Allocate a message buffer

Definition at line 751 of file mqttsink.c.

Here is the call graph for this function:



Here is the caller graph for this function:



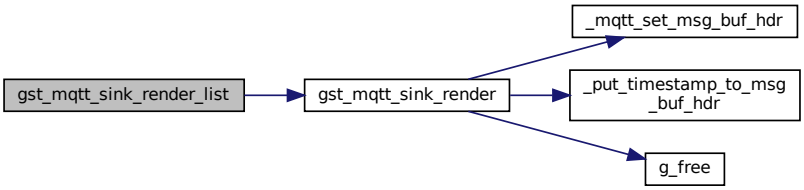
9.18.4.35 gst_mqtt_sink_render_list()

```
static GstFlowReturn gst_mqtt_sink_render_list (
            GstBaseSink * basesink,
            GstBufferList * list )  [static]
```

The callback to process GstBufferList (instead of a single buffer) on the sink pad.

Definition at line 877 of file mqttsink.c.

Here is the call graph for this function:



Here is the caller graph for this function:

9.18.4.36 gst_mqtt_sink_set_caps()

```
static gboolean gst_mqtt_sink_set_caps (
            GstBaseSink * basesink,
            GstCaps * caps )  [static]
```

An implementation of the set_caps vmethod in GstBaseSinkClass.

Definition at line 924 of file mqttsink.c.

Here is the call graph for this function:

Here is the caller graph for this function:

9.18.4.37 gst_mqtt_sink_set_client_id()

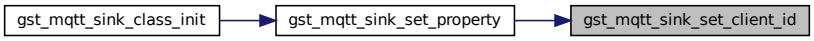
```
static void gst_mqtt_sink_set_client_id (
            GstMqttSink * self,
            const gchar * id )  [static]
```

Setter for the 'client-id' property.

Definition at line 985 of file mqttsink.c.

Here is the call graph for this function:

Here is the caller graph for this function:



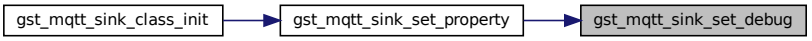
9.18.4.38 gst_mqtt_sink_set_debug()

```
static void gst_mqtt_sink_set_debug (
            GstMqttSink * self,
            const gboolean flag )  [static]
```

Setter for the 'debug' property.

Definition at line 967 of file mqttsink.c.

Here is the caller graph for this function:



9.18.4.39 gst_mqtt_sink_set_host_address()

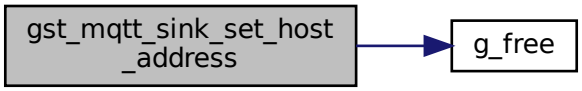
```
static void gst_mqtt_sink_set_host_address (
            GstMqttSink * self,
            const gchar * addr )  [static]
```

Setter for the 'host' property.

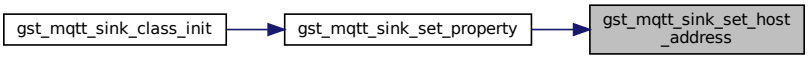
Todo Handle the case where the addr is changed at runtime

Definition at line 1004 of file mqttsink.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.18.4.40 gst_mqtt_sink_set_host_port()

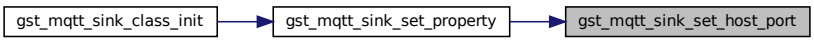
```
static void gst_mqtt_sink_set_host_port (
            GstMqttSink * self,
            const gchar * port )  [static]
```

Setter for the 'port' property.

Definition at line 1026 of file mqttsink.c.

Here is the call graph for this function:

Here is the caller graph for this function:



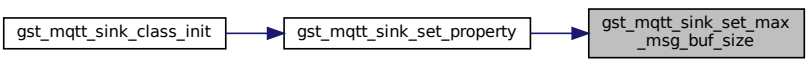
9.18.4.41 gst_mqtt_sink_set_max_msg_buf_size()

```
static void gst_mqtt_sink_set_max_msg_buf_size (
            GstMqttSink * self,
            const gsize size )  [static]
```

Setter for the 'max-buffer-size' property.

Definition at line 1118 of file mqttsink.c.

Here is the caller graph for this function:



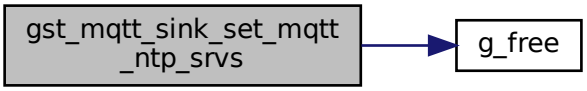
9.18.4.42 gst_mqtt_sink_set_mqtt_ntp_srvs()

```
static void gst_mqtt_sink_set_mqtt_ntp_srvs (
            GstMqttSink * self,
            const gchar * pairs )  [static]
```

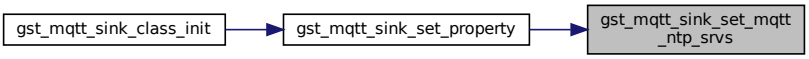
Setter for the 'ntp-srvs' property.

Definition at line 1194 of file mqttsink.c.

Here is the call graph for this function:



Here is the caller graph for this function:



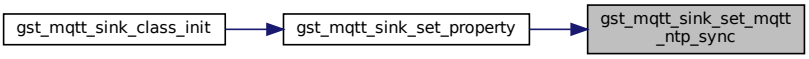
9.18.4.43 gst_mqtt_sink_set_mqtt_ntp_sync()

```
static void gst_mqtt_sink_set_mqtt_ntp_sync (
            GstMqttSink * self,
            const gboolean flag )  [static]
```

Setter for the 'ntp-sync' property.

Definition at line 1176 of file mqttsink.c.

Here is the caller graph for this function:



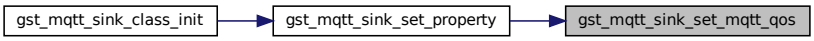
9.18.4.44 gst_mqtt_sink_set_mqtt_qos()

```
static void gst_mqtt_sink_set_mqtt_qos (
            GstMqttSink * self,
            const gint qos )  [static]
```

Setter for the 'mqtt-qos' property.

Definition at line 1158 of file mqttsink.c.

Here is the caller graph for this function:



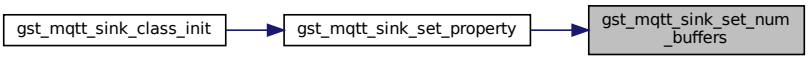
9.18.4.45 gst_mqtt_sink_set_num_buffers()

```
static void gst_mqtt_sink_set_num_buffers (
            GstMqttSink * self,
            const gint num )  [static]
```

Setter for the 'num-buffers' property.

Definition at line 1140 of file mqttsink.c.

Here is the caller graph for this function:



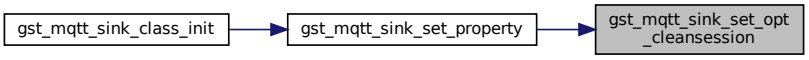
9.18.4.46 gst_mqtt_sink_set_opt_cleansession()

```
static void gst_mqtt_sink_set_opt_cleansession (
            GstMqttSink * self,
            const gboolean val )  [static]
```

Setter for the 'cleansession' property.

Definition at line 1064 of file mqttsink.c.

Here is the caller graph for this function:



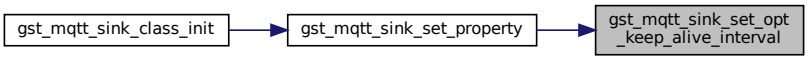
9.18.4.47 gst_mqtt_sink_set_opt_keep_alive_interval()

```
static void gst_mqtt_sink_set_opt_keep_alive_interval (
            GstMqttSink * self,
            const gint num )  [static]
```

Setter for the 'keep-alive-interval' property.

Definition at line 1100 of file mqttsink.c.

Here is the caller graph for this function:



9.18.4.48 gst_mqtt_sink_set_property()

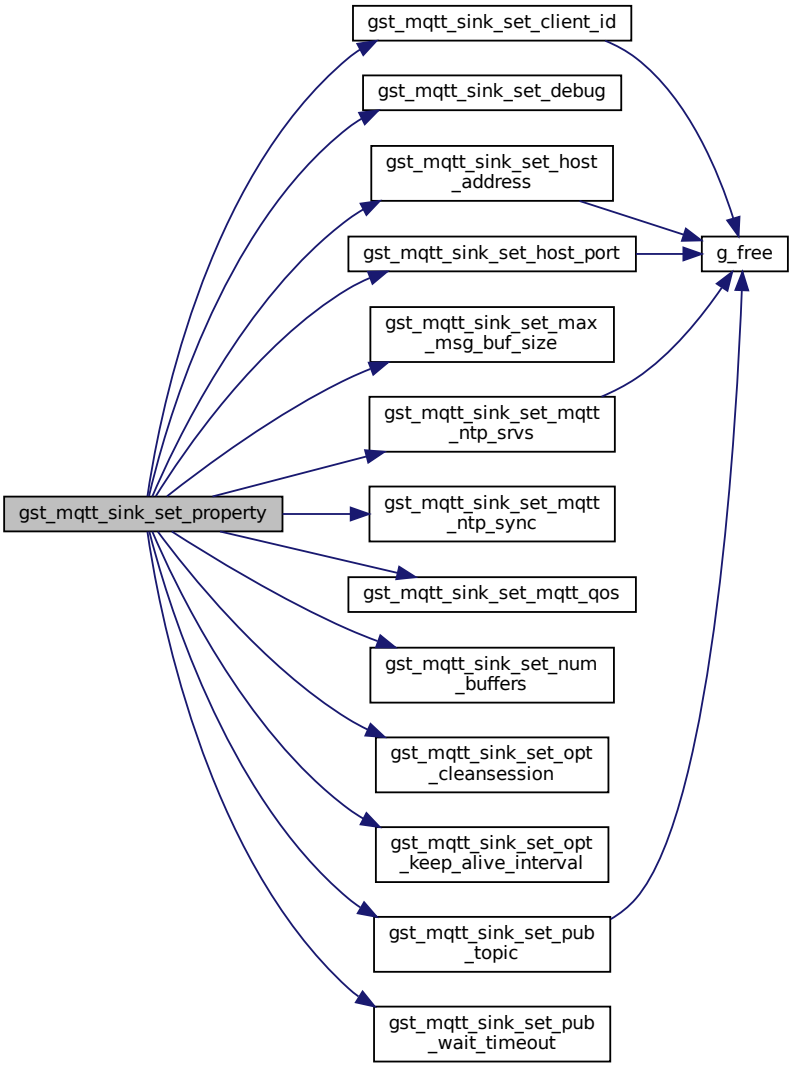
```
static void gst_mqtt_sink_set_property (
            GObject * object,
            guint prop_id,
            const GValue * value,
            GParamSpec * pspec )  [static]
```

The setter for the mqttsink's properties.

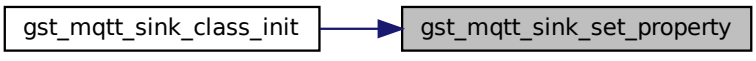
Function prototype declarations

Definition at line 344 of file mqttsink.c.

Here is the call graph for this function:



Here is the caller graph for this function:



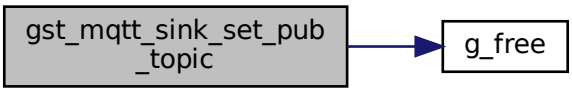
9.18.4.49 gst_mqtt_sink_set_pub_topic()

```
static void gst_mqtt_sink_set_pub_topic (
            GstMqttSink * self,
            const gchar * topic )  [static]
```

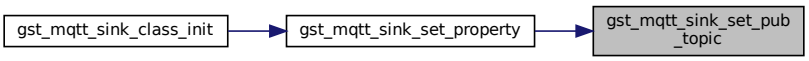
Setter for the 'pub-topic' property.

Definition at line 1045 of file mqttsink.c.

Here is the call graph for this function:



Here is the caller graph for this function:



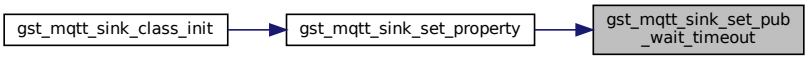
9.18.4.50 gst_mqtt_sink_set_pub_wait_timeout()

```
static void gst_mqtt_sink_set_pub_wait_timeout (
            GstMqttSink * self,
            const gulong to )  [static]
```

Setter for the 'pub-wait-timeout' property.

Definition at line 1082 of file mqttsink.c.

Here is the caller graph for this function:



9.18.4.51 gst_mqtt_sink_start()

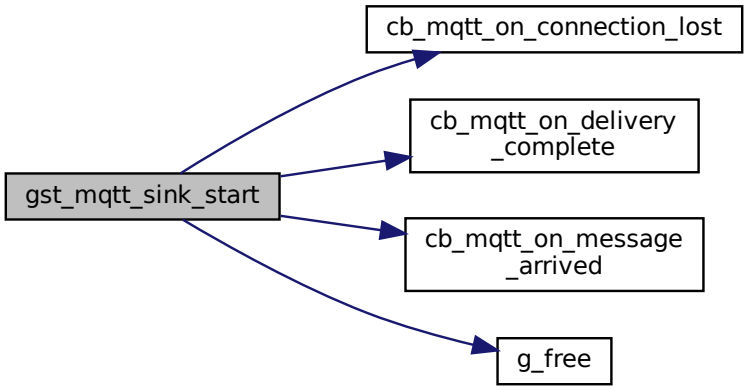
```
static gboolean gst_mqtt_sink_start (
            GstBaseSink * basesink )  [static]
```

Start mqttsink, called when state changed null to ready.

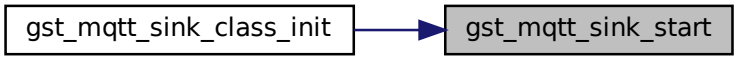
Todo Support other persistence mechanisms MQTTCLIENT_PERSISTENCE_NONE: A memory-based persistence mechanism MQTTCLIENT_PERSISTENCE_DEFAULT: The default file system-based persistence mechanism MQTTCLIENT_PERSISTENCE_USER: An application-specific persistence mechanism

Definition at line 554 of file mqttsink.c.

Here is the call graph for this function:



Here is the caller graph for this function:



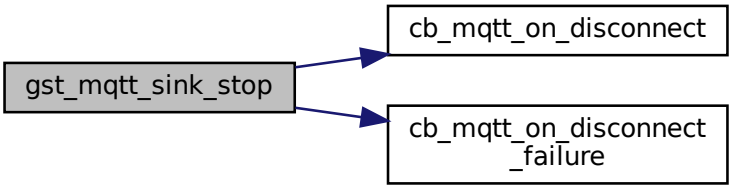
9.18.4.52 gst_mqtt_sink_stop()

```
static gboolean gst_mqtt_sink_stop (
            GstBaseSink * basesink )  [static]
```

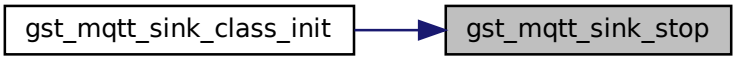
Stop mqttsink, called when state changed ready to null.

Definition at line 623 of file mqttsink.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.18.5 Variable Documentation

9.18.5.1 DEFAULT_MQTT_CLIENT_ID

```
const gchar DEFAULT_MQTT_CLIENT_ID[] = "$HOST_$PID_^[0-9][0-9]?$|^255$"  [static]
```

Definition at line 85 of file mqttsink.c.

9.18.5.2 DEFAULT_MQTT_CLIENT_ID_FORMAT

```
const gchar DEFAULT_MQTT_CLIENT_ID_FORMAT[] = "%s_%u_sink%u"  [static]
```

Definition at line 86 of file mqttsink.c.

9.18.5.3 DEFAULT_MQTT_HOST_ADDRESS

```
const gchar DEFAULT_MQTT_HOST_ADDRESS[] = "127.0.0.1"  [static]
```

Definition at line 82 of file mqttsink.c.

9.18.5.4 DEFAULT_MQTT_HOST_PORT

```
const gchar DEFAULT_MQTT_HOST_PORT[] = "1883"  [static]
```

Definition at line 83 of file mqttsink.c.

9.18.5.5 DEFAULT_MQTT_NTP_SERVERS

```
const gchar DEFAULT_MQTT_NTP_SERVERS[] = "pool.ntp.org:123"  [static]
```

Definition at line 89 of file mqttsink.c.

9.18.5.6 DEFAULT_MQTT_PUB_TOPIC

```
const gchar DEFAULT_MQTT_PUB_TOPIC[] = "$client-id/topic"  [static]
```

Definition at line 87 of file mqttsink.c.

9.18.5.7 DEFAULT_MQTT_PUB_TOPIC_FORMAT

```
const gchar DEFAULT_MQTT_PUB_TOPIC_FORMAT[] = "%s/topic"  [static]
```

Definition at line 88 of file mqttsink.c.

9.18.5.8 sink_client_id

```
guint8 sink_client_id = 0  [static]
```

Definition at line 81 of file mqttsink.c.

9.18.5.9 sink_pad_template

```
GstStaticPadTemplate sink_pad_template  [static]
```

Initial value:

```
= GST_STATIC_PAD_TEMPLATE ("sink",
    GST_PAD_SINK, GST_PAD_ALWAYS, GST_STATIC_CAPS_ANY)
```

Definition at line 35 of file mqttsink.c.

9.18.5.10 TAG_ERR_MQTTSINK

```
const gchar TAG_ERR_MQTTSINK[] = "ERROR: MQTTSink"  [static]
```

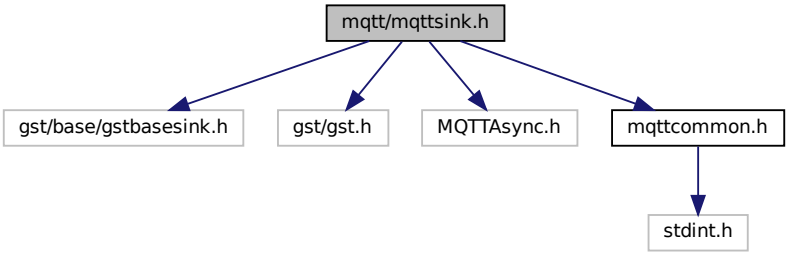
Definition at line 84 of file mqttsink.c.

9.19 mqtt/mqttsink.h File Reference

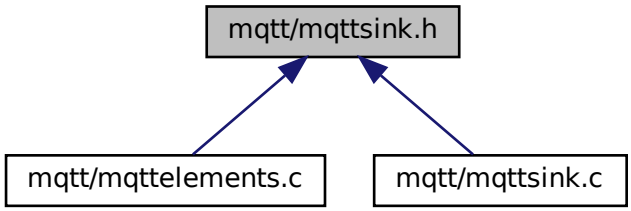
Publish incoming data streams as a MQTT topic.

```
#include <gst/base/gstbasesink.h>
#include <gst/gst.h>
#include <MQTTAsync.h>
#include "mqttcommon.h"
```

Include dependency graph for mqttsink.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct _GstMqttSink

  GstMqttSink data structure.
- struct _GstMqttSinkClass

  GstMqttSinkClass data structure.

Macros

- #define GST_TYPE_MQTT_SINK (gst_mqtt_sink_get_type())
- #define GST_MQTT_SINK(obj) (G_TYPE_CHECK_INSTANCE_CAST ((obj), GST_TYPE_MQTT_SINK, GstMqttSink))
- #define GST_IS_MQTT_SINK(obj) (G_TYPE_CHECK_INSTANCE_TYPE ((obj), GST_TYPE_MQTT_SINK))
- #define GST_MQTT_SINK_CAST(obj) ((GstMqttSink *) obj)
- #define GST_MQTT_SINK_CLASS(klass) (G_TYPE_CHECK_CLASS_CAST ((klass), GST_TYPE_MQTT_SINK, GstMqttSinkClass))
- #define GST_IS_MQTT_SINK_CLASS(klass) (G_TYPE_CHECK_CLASS_TYPE ((klass), GST_TYPE_MQTT_SINK))

Typedefs

- typedef struct _GstMqttSink GstMqttSink
- typedef struct _GstMqttSinkClass GstMqttSinkClass
- typedef enum _mqtt_sink_state_t mqtt_sink_state_t

  A type definition to indicate the state of this element.

Enumerations

- enum _mqtt_sink_state_t {
  MQTT_CONNECTION_LOST = -3, MQTT_CONNECT_FAILURE = -2, SINK_INITIALIZING = -1, SINK_RENDER_STOPPED = 0,
  SINK_RENDER_EOS, SINK_RENDER_ERROR, MQTT_CONNECTED, MQTT_DISCONNECTED,
  MQTT_DISCONNECT_FAILED }

  A type definition to indicate the state of this element.

Functions

- GType gst_mqtt_sink_get_type (void)

9.19.1 Detailed Description

Publish incoming data streams as a MQTT topic.

Copyright (C) 2021 Wook Song wook16.song@samsung.com

Date

08 Mar 2021

See also

https://github.com/nnstreamer/nnstreamer

Author

Wook Song wook16.song@samsung.com

Bug No known bugs except for NYI items

9.19.2 Macro Definition Documentation

9.19.2.1 GST_IS_MQTT_SINK

```
#define GST_IS_MQTT_SINK(
              obj ) (G_TYPE_CHECK_INSTANCE_TYPE ((obj), GST_TYPE_MQTT_SINK))
```

Definition at line 28 of file mqttsink.h.

9.19.2.2 GST_IS_MQTT_SINK_CLASS

```
#define GST_IS_MQTT_SINK_CLASS(
              klass ) (G_TYPE_CHECK_CLASS_TYPE ((klass), GST_TYPE_MQTT_SINK))
```

Definition at line 34 of file mqttsink.h.

9.19.2.3 GST_MQTT_SINK

```
#define GST_MQTT_SINK(
              obj ) (G_TYPE_CHECK_INSTANCE_CAST ((obj), GST_TYPE_MQTT_SINK, GstMqttSink))
```

Definition at line 26 of file mqttsink.h.

9.19.2.4 GST_MQTT_SINK_CAST

```
#define GST_MQTT_SINK_CAST(
              obj ) ((GstMqttSink *) obj)
```

Definition at line 30 of file mqttsink.h.

9.19.2.5 GST_MQTT_SINK_CLASS

```
#define GST_MQTT_SINK_CLASS(
              klass ) (G_TYPE_CHECK_CLASS_CAST ((klass), GST_TYPE_MQTT_SINK, GstMqttSinkClass))
```

Definition at line 32 of file mqttsink.h.

9.19.2.6 GST_TYPE_MQTT_SINK

```
#define GST_TYPE_MQTT_SINK (gst_mqtt_sink_get_type())
```

Definition at line 24 of file mqttsink.h.

9.19.3 Typedef Documentation

9.19.3.1 GstMqttSink

```
typedef struct _GstMqttSink GstMqttSink
```

Definition at line 37 of file mqttsink.h.

9.19.3.2 GstMqttSinkClass

```
typedef struct _GstMqttSinkClass GstMqttSinkClass
```

Definition at line 38 of file mqttsink.h.

9.19.3.3 mqtt_sink_state_t

```
typedef enum _mqtt_sink_state_t mqtt_sink_state_t
```

A type definition to indicate the state of this element.

9.19.4 Enumeration Type Documentation

9.19.4.1 _mqtt_sink_state_t

```
enum _mqtt_sink_state_t
```

A type definition to indicate the state of this element.

Enumerator

MQTT_CONNECTION_LOST	
MQTT_CONNECT_FAILURE	
SINK_INITIALIZING	
SINK_RENDER_STOPPED	
SINK_RENDER_EOS	
SINK_RENDER_ERROR	
MQTT_CONNECTED	
MQTT_DISCONNECTED	
MQTT_DISCONNECT_FAILED	

Definition at line 43 of file mqttsink.h.

9.19.5 Function Documentation

9.19.5.1 gst_mqtt_sink_get_type()

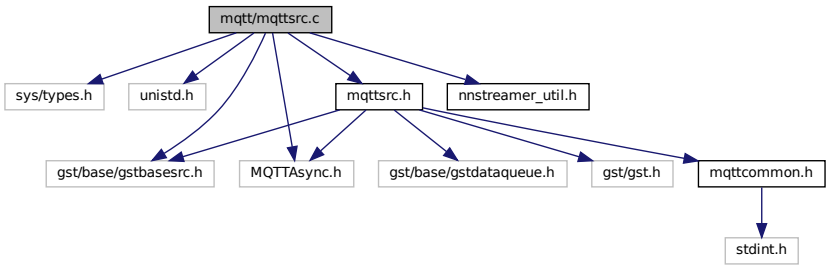
```
GType gst_mqtt_sink_get_type (
            void  )
```

9.20 mqtt/mqttsrc.c File Reference

Subscribe a MQTT topic and push incoming data to the GStreamer pipeline.

```
#include <sys/types.h>
#include <unistd.h>
#include <gst/base/gstbasesrc.h>
#include <MQTTAsync.h>
#include <nnstreamer_util.h>
#include "mqttsrc.h"
```

Include dependency graph for mqttsrc.c:



Macros

- #define gst_mqtt_src_parent_class parent_class
- #define GST_CAT_DEFAULT gst_mqtt_src_debug

Enumerations

- enum {
  PROP_0, PROP_DEBUG, PROP_IS_LIVE, PROP_MQTT_CLIENT_ID,
  PROP_MQTT_HOST_ADDRESS, PROP_MQTT_HOST_PORT, PROP_MQTT_SUB_TOPIC, PROP_MQTT_SUB_TIMEOUT
  PROP_MQTT_OPT_CLEANSESSION, PROP_MQTT_OPT_KEEP_ALIVE_INTERVAL, PROP_MQTT_QOS,
  PROP_LAST }
- enum {
  DEFAULT_DEBUG = FALSE, DEFAULT_IS_LIVE = TRUE, DEFAULT_MQTT_OPT_CLEANSESSION =
  TRUE, DEFAULT_MQTT_OPT_KEEP_ALIVE_INTERVAL = 60,
  DEFAULT_MQTT_SUB_TIMEOUT = 10000000, DEFAULT_MQTT_SUB_TIMEOUT_MIN = 1000000,
  DEFAULT_MQTT_QOS = 2 }

Functions

- G_DEFINE_TYPE (GstMqttSrc, gst_mqtt_src, GST_TYPE_BASE_SRC)
- GST_DEBUG_CATEGORY_STATIC (gst_mqtt_src_debug)
- static void gst_mqtt_src_set_property (GObject *object, guint prop_id, const GValue *value, GParamSpec *pspec)

  The setter for the mqttsrc's properties.
- static void gst_mqtt_src_get_property (GObject *object, guint prop_id, GValue *value, GParamSpec *pspec)

  The getter for the mqttsrc's properties.

- static void gst_mqtt_src_class_finalize (GObject *object)

  Finalize GstMqttSrcClass object.
- static GstStateChangeReturn gst_mqtt_src_change_state (GstElement *element, GstStateChange transition)

  Handle mqttsrc's state change.
- static gboolean gst_mqtt_src_start (GstBaseSrc *basesrc)

  Start mqttsrc, called when state changed null to ready.
- static gboolean gst_mqtt_src_stop (GstBaseSrc *basesrc)

  Stop mqttsrc, called when state changed ready to null.
- static GstCaps * gst_mqtt_src_get_caps (GstBaseSrc *basesrc, GstCaps *filter)

  Get caps of subclass.
- static gboolean gst_mqtt_src_renegotiate (GstBaseSrc *basesrc)

  Do negotiation procedure again if it needed.
- static void gst_mqtt_src_get_times (GstBaseSrc *basesrc, GstBuffer *buffer, GstClockTime *start, GstClockTime *end)

  Return the time information of the given buffer.
- static gboolean gst_mqtt_src_is_seekable (GstBaseSrc *basesrc)

  Check if source supports seeking.
- static GstFlowReturn gst_mqtt_src_create (GstBaseSrc *basesrc, guint64 offset, guint size, GstBuffer **buf)

  Create a buffer containing the subscribed data.
- static gboolean gst_mqtt_src_query (GstBaseSrc *basesrc, GstQuery *query)

  An implementation of the GstBaseSrc vmethod that handles queries.
- static gboolean gst_mqtt_src_get_debug (GstMqttSrc *self)

  Getter for the 'debug' property.
- static void gst_mqtt_src_set_debug (GstMqttSrc *self, const gboolean flag)

  Setter for the 'debug' property.
- static gboolean gst_mqtt_src_get_is_live (GstMqttSrc *self)

  Getter for the 'is-live' property.
- static void gst_mqtt_src_set_is_live (GstMqttSrc *self, const gboolean flag)

  Setter for the 'is-live' property.
- static gchar * gst_mqtt_src_get_client_id (GstMqttSrc *self)

  Getter for the 'client-id' property.
- static void gst_mqtt_src_set_client_id (GstMqttSrc *self, const gchar *id)

  Setter for the 'client-id' property.
- static gchar * gst_mqtt_src_get_host_address (GstMqttSrc *self)

  Getter for the 'host' property.
- static void gst_mqtt_src_set_host_address (GstMqttSrc *self, const gchar *addr)

  Setter for the 'host' property.
- static gchar * gst_mqtt_src_get_host_port (GstMqttSrc *self)

  Getter for the 'port' property.
- static void gst_mqtt_src_set_host_port (GstMqttSrc *self, const gchar *port)

  Setter for the 'port' property.
- static gint64 gst_mqtt_src_get_sub_timeout (GstMqttSrc *self)

  Getter for the 'sub-timeout' property.
- static void gst_mqtt_src_set_sub_timeout (GstMqttSrc *self, const gint64 t)

  Setter for the 'sub-timeout' property.
- static gchar * gst_mqtt_src_get_sub_topic (GstMqttSrc *self)

  Getter for the 'sub-topic' property.
- static void gst_mqtt_src_set_sub_topic (GstMqttSrc *self, const gchar *topic)

  Setter for the 'sub-topic' property.
- static gboolean gst_mqtt_src_get_opt_cleansession (GstMqttSrc *self)

Getter for the 'cleansession' property.

- static void gst_mqtt_src_set_opt_cleansession (GstMqttSrc *self, const gboolean val)

  Setter for the 'cleansession' property.

- static gint gst_mqtt_src_get_opt_keep_alive_interval (GstMqttSrc *self)

  Getter for the 'keep-alive-interval' property.

- static void gst_mqtt_src_set_opt_keep_alive_interval (GstMqttSrc *self, const gint num)

  Setter for the 'keep-alive-interval' property.

- static gint gst_mqtt_src_get_mqtt_qos (GstMqttSrc *self)

  Getter for the 'mqtt-qos' property.

- static void gst_mqtt_src_set_mqtt_qos (GstMqttSrc *self, const gint qos)

  Setter for the 'mqtt-qos' property.

- static void cb_mqtt_on_connection_lost (void *context, char *cause)

  A callback to handle the connection lost to the broker.

- static int cb_mqtt_on_message_arrived (void *context, char *topic_name, int topic_len, MQTTAsync_↩ message *message)

  A callback to handle the arrived message.

- static void cb_mqtt_on_connect (void *context, MQTTAsync_successData *response)

  A callback invoked when the connection is established.

- static void cb_mqtt_on_connect_failure (void *context, MQTTAsync_failureData *response)

  A callback invoked when it is failed to connect to the broker.

- static void cb_mqtt_on_subscribe (void *context, MQTTAsync_successData *response)

  MQTTAsync_responseOptions's onSuccess callback for MQTTAsync_subscribe ()

- static void cb_mqtt_on_subscribe_failure (void *context, MQTTAsync_failureData *response)

  MQTTAsync_responseOptions's onFailure callback for MQTTAsync_subscribe ()

- static void cb_mqtt_on_unsubscribe (void *context, MQTTAsync_successData *response)

  MQTTAsync_responseOptions's onSuccess callback for MQTTAsync_unsubscribe ()

- static void cb_mqtt_on_unsubscribe_failure (void *context, MQTTAsync_failureData *response)

  MQTTAsync_responseOptions's onFailure callback for MQTTAsync_unsubscribe ()

- static void cb_memory_wrapped_destroy (void *p)

  A callback invoked when destroying the GstMemory which wrapped the arrived message.

- static GstMQTTMessageHdr * _extract_mqtt_msg_hdr_from (GstMemory *mem, GstMemory **hdr_mem, GstMapInfo *hdr_map_info)

  A utility function to extract header information from a received message.

- static void _put_timestamp_on_gst_buf (GstMqttSrc *self, GstMQTTMessageHdr *hdr, GstBuffer *buf)

  A utility function to put the timestamp information onto a GstBuffer-typed buffer using the given packet header.

- static gboolean _subscribe (GstMqttSrc *self)

  A helper function to properly invoke MQTTAsync_subscribe ()

- static gboolean _unsubscribe (GstMqttSrc *self)

  A wrapper function that calls MQTTAsync_unsubscribe ()

- static gboolean _is_gst_buffer_timestamp_valid (GstBuffer *buf)

  A utility function to check whether the timestamp marked by _put_timestamp_on_gst_buf () is valid or not.

- static void gst_mqtt_src_init (GstMqttSrc *self)

  Initialize GstMqttSrc object.

- static void gst_mqtt_src_class_init (GstMqttSrcClass *klass)

  Initialize GstMqttSrcClass object.

Variables

- static GstStaticPadTemplate src_pad_template
- static guint8 src_client_id = 0
- static const gchar DEFAULT_MQTT_HOST_ADDRESS [ ] = "127.0.0.1"
- static const gchar DEFAULT_MQTT_HOST_PORT [ ] = "1883"
- static const gchar TAG_ERR_MQTTSRC [ ] = "ERROR: MQTTSrc"
- static const gchar DEFAULT_MQTT_CLIENT_ID [ ]
- static const gchar DEFAULT_MQTT_CLIENT_ID_FORMAT [ ] = "%s_%u_src%u"

9.20.1 Detailed Description

Subscribe a MQTT topic and push incoming data to the GStreamer pipeline.

Copyright (C) 2021 Wook Song wook16.song@samsung.com

Date

08 Mar 2021

See also

https://github.com/nnstreamer/nnstreamer

Author

Wook Song wook16.song@samsung.com

Bug No known bugs except for NYI items

9.20.2 Macro Definition Documentation

9.20.2.1 GST_CAT_DEFAULT

```
#define GST_CAT_DEFAULT gst_mqtt_src_debug
```

Definition at line 38 of file mqttsrc.c.

9.20.2.2 gst_mqtt_src_parent_class

```
#define gst_mqtt_src_parent_class parent_class
```

Definition at line 34 of file mqttsrc.c.

9.20.3 Enumeration Type Documentation

9.20.3.1 anonymous enum

```
anonymous enum
```

Enumerator

PROP_0	
PROP_DEBUG	
PROP_IS_LIVE	
PROP_MQTT_CLIENT_ID	
PROP_MQTT_HOST_ADDRESS	
PROP_MQTT_HOST_PORT	
PROP_MQTT_SUB_TOPIC	
PROP_MQTT_SUB_TIMEOUT	
PROP_MQTT_OPT_CLEANSESSION	
PROP_MQTT_OPT_KEEP_ALIVE_INTERVAL	
PROP_MQTT_QOS	
PROP_LAST	

Definition at line 40 of file mqttsrc.c.

9.20.3.2 anonymous enum

```
anonymous enum
```

Enumerator

DEFAULT_DEBUG	
DEFAULT_IS_LIVE	
DEFAULT_MQTT_OPT_CLEANSESSION	
DEFAULT_MQTT_OPT_KEEP_ALIVE_INTERVAL	
DEFAULT_MQTT_SUB_TIMEOUT	
DEFAULT_MQTT_SUB_TIMEOUT_MIN	
DEFAULT_MQTT_QOS	

Definition at line 58 of file mqttsrc.c.

9.20.4 Function Documentation

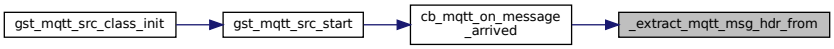
9.20.4.1 _extract_mqtt_msg_hdr_from()

```
static GstMQTTMessageHdr * _extract_mqtt_msg_hdr_from (
            GstMemory * mem,
            GstMemory ** hdr_mem,
            GstMapInfo * hdr_map_info )  [static]
```

A utility function to extract header information from a received message.

Definition at line 1347 of file mqttsrc.c.

Here is the caller graph for this function:



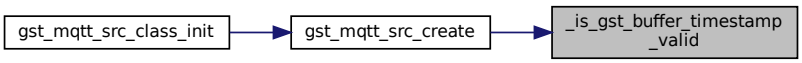
9.20.4.2 _is_gst_buffer_timestamp_valid()

```
static gboolean _is_gst_buffer_timestamp_valid (
            GstBuffer * buf )  [inline], [static]
```

A utility function to check whether the timestamp marked by _put_timestamp_on_gst_buf () is valid or not.

Definition at line 156 of file mqttsrc.c.

Here is the caller graph for this function:



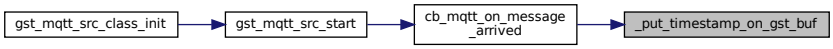
9.20.4.3 _put_timestamp_on_gst_buf()

```
static void _put_timestamp_on_gst_buf (
            GstMqttSrc * self,
            GstMQTTMessageHdr * hdr,
            GstBuffer * buf )  [static]
```

A utility function to put the timestamp information onto a GstBuffer-typed buffer using the given packet header.

Definition at line 1366 of file mqttsrc.c.

Here is the caller graph for this function:



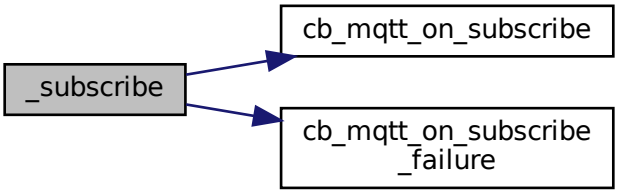
9.20.4.4 _subscribe()

```
static gboolean _subscribe (
            GstMqttSrc * self )  [static]
```

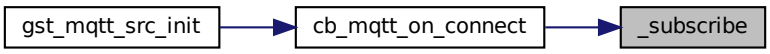
A helper function to properly invoke MQTTAsync_subscribe ()

Definition at line 1308 of file mqttsrc.c.

Here is the call graph for this function:



Here is the caller graph for this function:



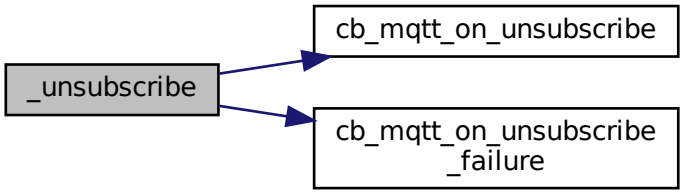
9.20.4.5 _unsubscribe()

```
static gboolean _unsubscribe (
            GstMqttSrc * self )  [static]
```

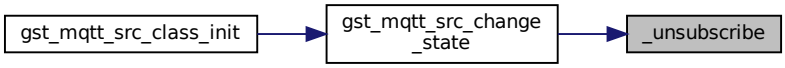
A wrapper function that calls MQTTAsync_unsubscribe ()

Definition at line 1328 of file mqttsrc.c.

Here is the call graph for this function:



Here is the caller graph for this function:



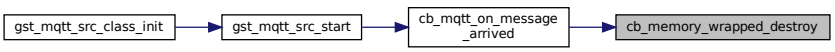
9.20.4.6 cb_memory_wrapped_destroy()

```
static void cb_memory_wrapped_destroy (
            void * p )  [static]
```

A callback invoked when destroying the GstMemory which wrapped the arrived message.

Definition at line 1179 of file mqttsrc.c.

Here is the caller graph for this function:



9.20.4.7 cb_mqtt_on_connect()

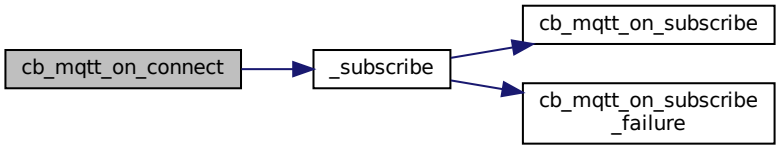
```
static void cb_mqtt_on_connect (
            void * context,
            MQTTAsync_successData * response )  [static]
```

A callback invoked when the connection is established.

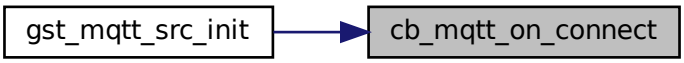
GstFlowReturn is an enum type. It is possible to use int here

Definition at line 1190 of file mqttsrc.c.

Here is the call graph for this function:



Here is the caller graph for this function:



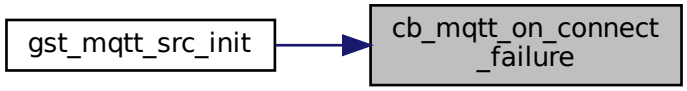
9.20.4.8 cb_mqtt_on_connect_failure()

```
static void cb_mqtt_on_connect_failure (
            void * context,
            MQTTAsync_failureData * response )  [static]
```

A callback invoked when it is failed to connect to the broker.

Definition at line 1223 of file mqttsrc.c.

Here is the caller graph for this function:



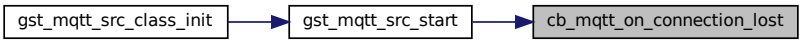
9.20.4.9 cb_mqtt_on_connection_lost()

```
static void cb_mqtt_on_connection_lost (
            void * context,
            char * cause )  [static]
```

A callback to handle the connection lost to the broker.

Definition at line 1053 of file mqttsrc.c.

Here is the caller graph for this function:



9.20.4.10 cb_mqtt_on_message_arrived()

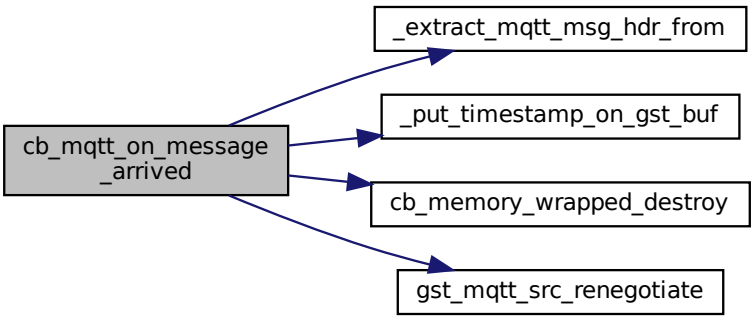
```
static int cb_mqtt_on_message_arrived (
            void * context,
            char * topic_name,
            int topic_len,
            MQTTAsync_message * message )  [static]
```

A callback to handle the arrived message.

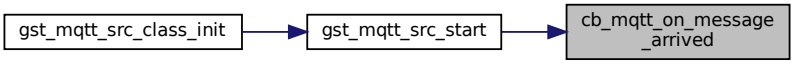
Timestamp synchronization

Definition at line 1075 of file mqttsrc.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.20.4.11 cb_mqtt_on_subscribe()

```
static void cb_mqtt_on_subscribe (
            void * context,
            MQTTAsync_successData * response )  [static]
```

MQTTAsync_responseOptions's onSuccess callback for MQTTAsync_subscribe ()

Definition at line 1242 of file mqttsrc.c.

Here is the caller graph for this function:

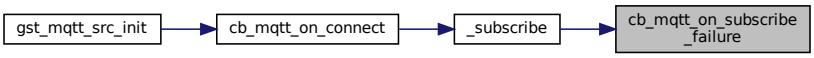
9.20.4.12 cb_mqtt_on_subscribe_failure()

```
static void cb_mqtt_on_subscribe_failure (
            void * context,
            MQTTAsync_failureData * response )  [static]
```

MQTTAsync_responseOptions's onFailure callback for MQTTAsync_subscribe ()

Definition at line 1257 of file mqttsrc.c.

Here is the caller graph for this function:



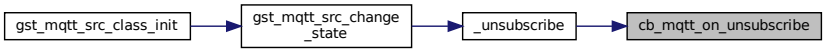
9.20.4.13 cb_mqtt_on_unsubscribe()

```
static void cb_mqtt_on_unsubscribe (
            void * context,
            MQTTAsync_successData * response )  [static]
```

MQTTAsync_responseOptions's onSuccess callback for MQTTAsync_unsubscribe ()

Definition at line 1275 of file mqttsrc.c.

Here is the caller graph for this function:



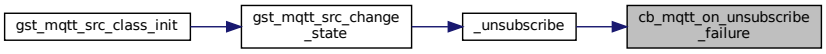
9.20.4.14 cb_mqtt_on_unsubscribe_failure()

```
static void cb_mqtt_on_unsubscribe_failure (
            void * context,
            MQTTAsync_failureData * response )  [static]
```

MQTTAsync_responseOptions's onFailure callback for MQTTAsync_unsubscribe ()

Definition at line 1290 of file mqttsrc.c.

Here is the caller graph for this function:



9.20.4.15 G_DEFINE_TYPE()

```
G_DEFINE_TYPE (
             GstMqttSrc ,
             gst_mqtt_src ,
             GST_TYPE_BASE_SRC  )
```

9.20.4.16 GST_DEBUG_CATEGORY_STATIC()

```
GST_DEBUG_CATEGORY_STATIC (
             gst_mqtt_src_debug  )
```

9.20.4.17 gst_mqtt_src_change_state()

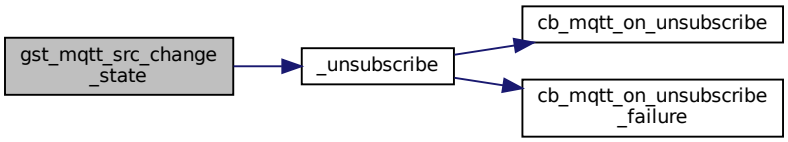
```
static GstStateChangeReturn gst_mqtt_src_change_state (
             GstElement * element,
             GstStateChange transition )  [static]
```

Handle mqttsrc's state change.

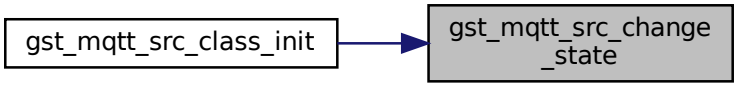
This handles the case when the state is changed to PLAYING again

Definition at line 440 of file mqttsrc.c.

Here is the call graph for this function:



Here is the caller graph for this function:



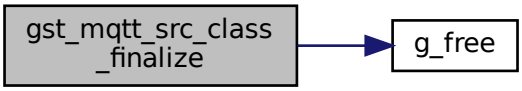
9.20.4.18 gst_mqtt_src_class_finalize()

```
static void gst_mqtt_src_class_finalize (
            GObject * object )  [static]
```

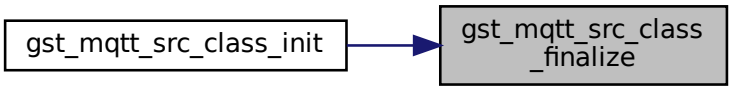
Finalize GstMqttSrcClass object.

Definition at line 408 of file mqttsrc.c.

Here is the call graph for this function:



Here is the caller graph for this function:



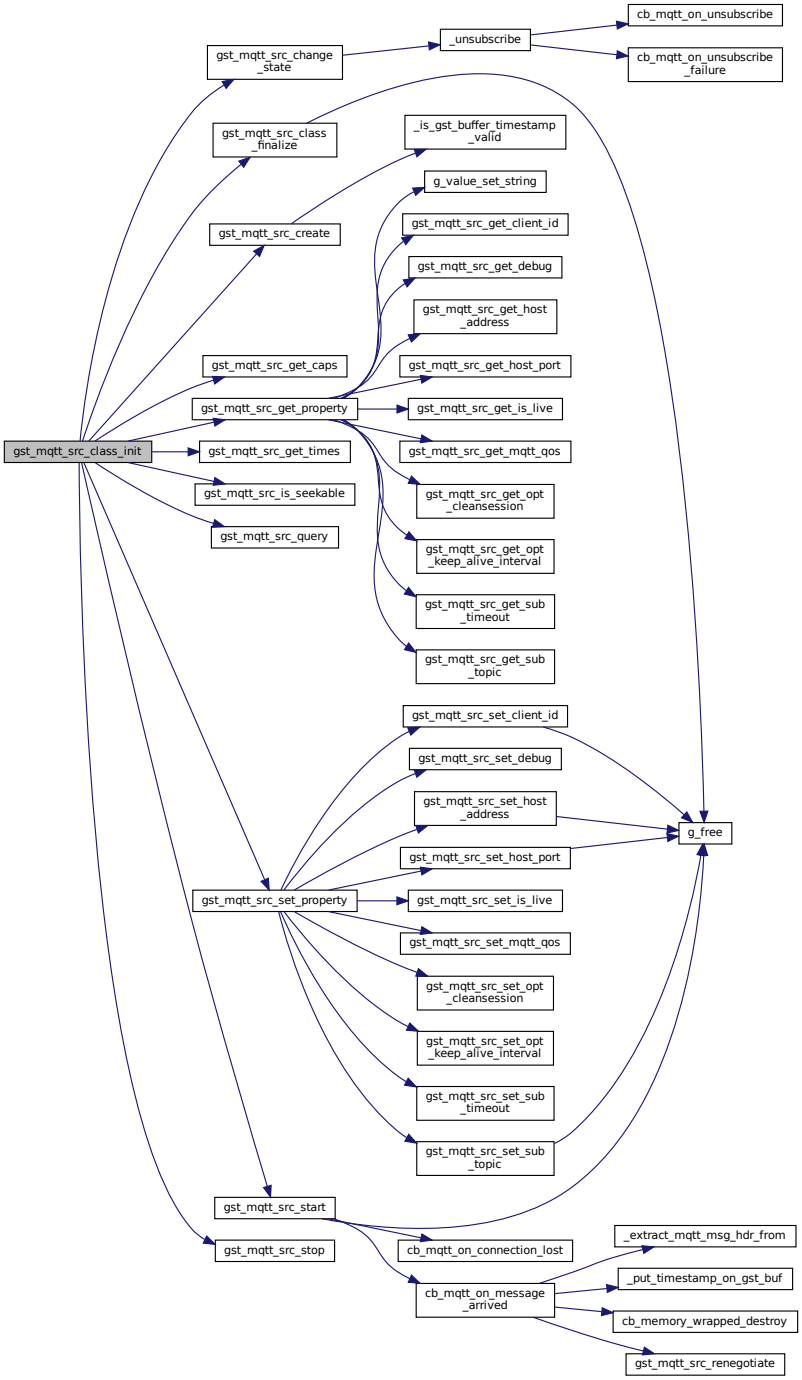
9.20.4.19 gst_mqtt_src_class_init()

```
static void gst_mqtt_src_class_init (
            GstMqttSrcClass * klass )  [static]
```

Initialize GstMqttSrcClass object.

Definition at line 222 of file mqttsrc.c.

Here is the call graph for this function:



9.20.4.20 gst_mqtt_src_create()

```
static GstFlowReturn gst_mqtt_src_create (
            GstBaseSrc * basesrc,
```

```
guint64 offset,
guint size,
GstBuffer ** buf )  [static]
```

Create a buffer containing the subscribed data.

Todo DEFAULT_MQTT_SUB_TIMEOUT_MIN is too long

This buffer is coming from the past. Drop it.

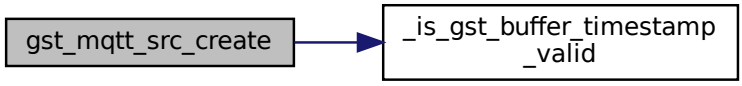
Update latency

Todo If the difference between new latency and old latency, gst_element_post_message (GST_ELEMENT_CAST (self), gst_message_new_latency (GST_OBJECT_CAST (self))); is needed.

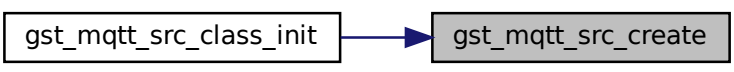
Todo : Send EoS here

Definition at line 710 of file mqttsrc.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.20.4.21 gst_mqtt_src_get_caps()

```
static GstCaps * gst_mqtt_src_get_caps (
            GstBaseSrc * basesrc,
            GstCaps * filter )  [static]
```

Get caps of subclass.

Definition at line 601 of file mqttsrc.c.

Here is the caller graph for this function:

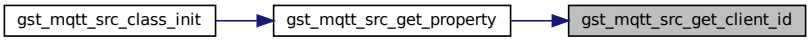
9.20.4.22 gst_mqtt_src_get_client_id()

```
static gchar * gst_mqtt_src_get_client_id (
            GstMqttSrc * self )  [static]
```

Getter for the 'client-id' property.

Definition at line 902 of file mqttsrc.c.

Here is the caller graph for this function:



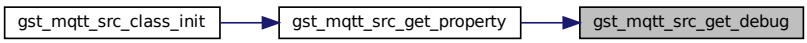
9.20.4.23 gst_mqtt_src_get_debug()

```
static gboolean gst_mqtt_src_get_debug (
            GstMqttSrc * self )  [static]
```

Getter for the 'debug' property.

Definition at line 865 of file mqttsrc.c.

Here is the caller graph for this function:



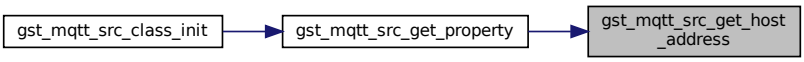
9.20.4.24 gst_mqtt_src_get_host_address()

```
static gchar * gst_mqtt_src_get_host_address (
            GstMqttSrc * self )  [static]
```

Getter for the 'host' property.

Definition at line 921 of file mqttsrc.c.

Here is the caller graph for this function:



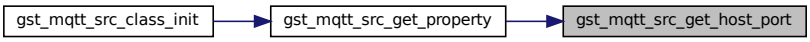
9.20.4.25 gst_mqtt_src_get_host_port()

```
static gchar * gst_mqtt_src_get_host_port (
            GstMqttSrc * self )  [static]
```

Getter for the 'port' property.

Definition at line 943 of file mqttsrc.c.

Here is the caller graph for this function:



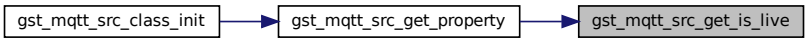
9.20.4.26 gst_mqtt_src_get_is_live()

```
static gboolean gst_mqtt_src_get_is_live (
            GstMqttSrc * self )  [static]
```

Getter for the 'is-live' property.

Definition at line 883 of file mqttsrc.c.

Here is the caller graph for this function:



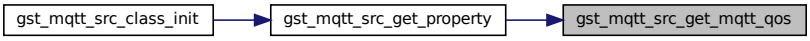
9.20.4.27 gst_mqtt_src_get_mqtt_qos()

```
static gint gst_mqtt_src_get_mqtt_qos (
            GstMqttSrc * self )  [static]
```

Getter for the 'mqtt-qos' property.

Definition at line 1035 of file mqttsrc.c.

Here is the caller graph for this function:



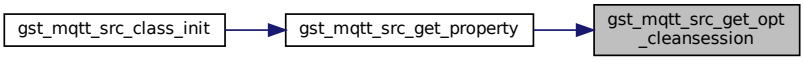
9.20.4.28 gst_mqtt_src_get_opt_cleansession()

```
static gboolean gst_mqtt_src_get_opt_cleansession (
            GstMqttSrc * self )  [static]
```

Getter for the 'cleansession' property.

Definition at line 999 of file mqttsrc.c.

Here is the caller graph for this function:



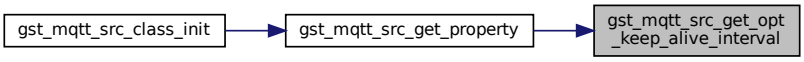
9.20.4.29 gst_mqtt_src_get_opt_keep_alive_interval()

```
static gint gst_mqtt_src_get_opt_keep_alive_interval (
            GstMqttSrc * self )  [static]
```

Getter for the 'keep-alive-interval' property.

Definition at line 1017 of file mqttsrc.c.

Here is the caller graph for this function:



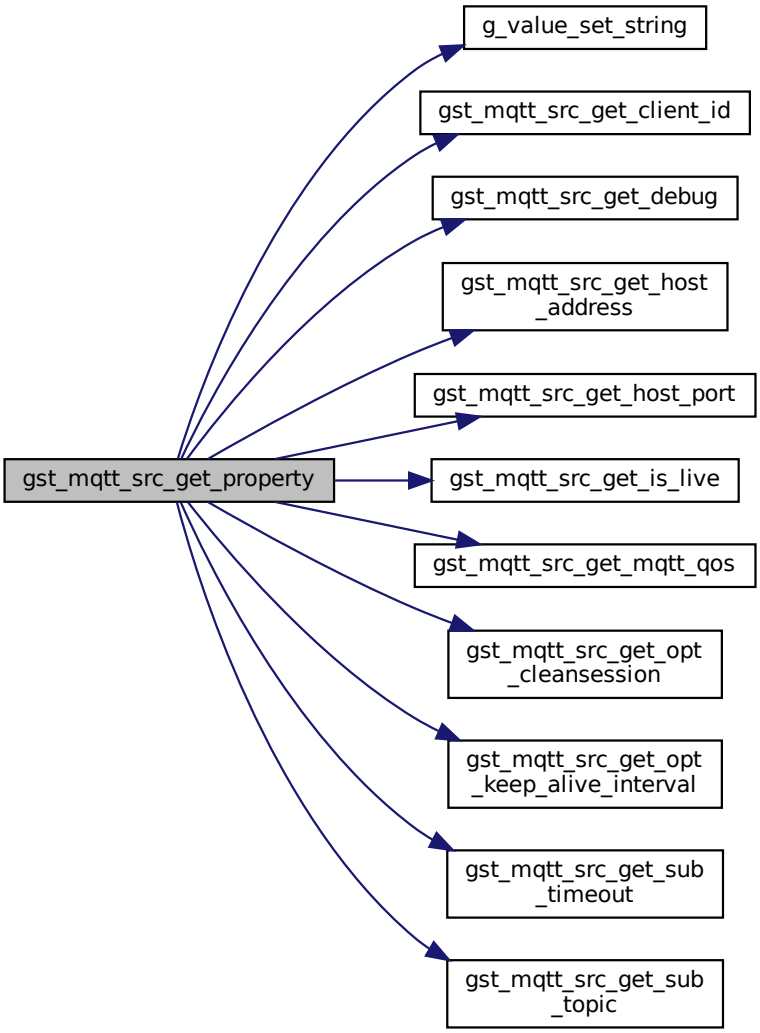
9.20.4.30 gst_mqtt_src_get_property()

```
static void gst_mqtt_src_get_property (
            GObject * object,
            guint prop_id,
            GValue * value,
            GParamSpec * pspec )  [static]
```

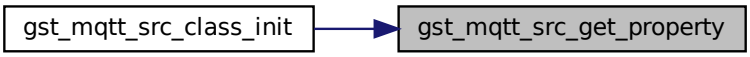
The getter for the mqttsrc's properties.

Definition at line 362 of file mqttsrc.c.

Here is the call graph for this function:



Here is the caller graph for this function:



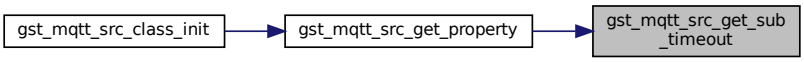
9.20.4.31 gst_mqtt_src_get_sub_timeout()

```
static gint64 gst_mqtt_src_get_sub_timeout (
            GstMqttSrc * self )  [static]
```

Getter for the 'sub-timeout' property.

Definition at line 962 of file mqttsrc.c.

Here is the caller graph for this function:



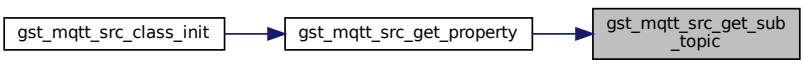
9.20.4.32 gst_mqtt_src_get_sub_topic()

```
static gchar * gst_mqtt_src_get_sub_topic (
            GstMqttSrc * self )  [static]
```

Getter for the 'sub-topic' property.

Definition at line 980 of file mqttsrc.c.

Here is the caller graph for this function:



9.20.4.33 gst_mqtt_src_get_times()

```
static void gst_mqtt_src_get_times (
            GstBaseSrc * basesrc,
            GstBuffer * buffer,
            GstClockTime * start,
            GstClockTime * end )  [static]
```

Return the time information of the given buffer.

Definition at line 674 of file mqttsrc.c.

Here is the caller graph for this function:

9.20.4.34 gst_mqtt_src_init()

```
static void gst_mqtt_src_init (
            GstMqttSrc * self )  [static]
```

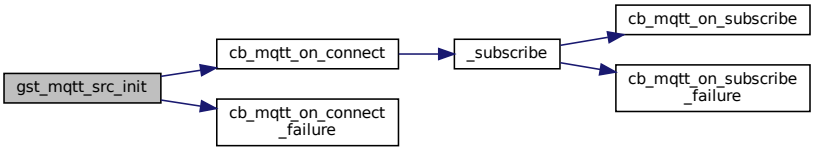
Initialize GstMqttSrc object.

Function definitions init mqttsrc properties

init private member variables

Definition at line 169 of file mqttsrc.c.

Here is the call graph for this function:



9.20.4.35 gst_mqtt_src_is_seekable()

```
static gboolean gst_mqtt_src_is_seekable (
            GstBaseSrc * basesrc )  [static]
```

Check if source supports seeking.

Note

Seeking is not supported since this element handles live subscription data.

Definition at line 700 of file mqttsrc.c.

Here is the caller graph for this function:

9.20.4.36 gst_mqtt_src_query()

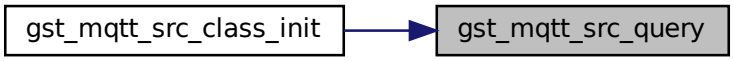
```
static gboolean gst_mqtt_src_query (
            GstBaseSrc * basesrc,
            GstQuery * query )  [static]
```

An implementation of the GstBaseSrc vmethod that handles queries.

The second argument of gst_query_set_latency should be always TRUE.

Definition at line 819 of file mqttsrc.c.

Here is the caller graph for this function:



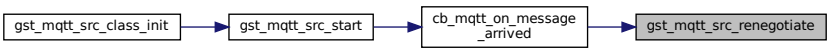
9.20.4.37 gst_mqtt_src_renegotiate()

```
static gboolean gst_mqtt_src_renegotiate (
            GstBaseSrc * basesrc )  [static]
```

Do negotiation procedure again if it needed.

Definition at line 624 of file mqttsrc.c.

Here is the caller graph for this function:



9.20.4.38 gst_mqtt_src_set_client_id()

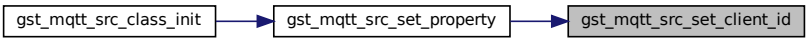
```
static void gst_mqtt_src_set_client_id (
            GstMqttSrc * self,
            const gchar * id )  [static]
```

Setter for the 'client-id' property.

Definition at line 911 of file mqttsrc.c.

Here is the call graph for this function:

Here is the caller graph for this function:



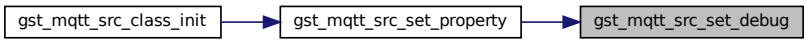
9.20.4.39 gst_mqtt_src_set_debug()

```
static void gst_mqtt_src_set_debug (
            GstMqttSrc * self,
            const gboolean flag )  [static]
```

Setter for the 'debug' property.

Definition at line 874 of file mqttsrc.c.

Here is the caller graph for this function:



9.20.4.40 gst_mqtt_src_set_host_address()

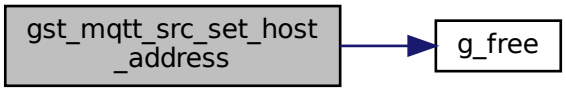
```
static void gst_mqtt_src_set_host_address (
            GstMqttSrc * self,
            const gchar * addr )  [static]
```

Setter for the 'host' property.

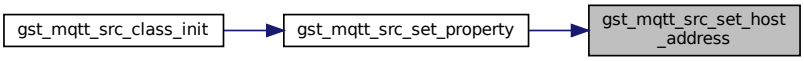
Todo Handle the case where the addr is changed at runtime

Definition at line 930 of file mqttsrc.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.20.4.41 gst_mqtt_src_set_host_port()

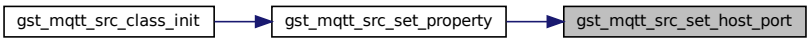
```
static void gst_mqtt_src_set_host_port (
            GstMqttSrc * self,
            const gchar * port )  [static]
```

Setter for the 'port' property.

Definition at line 952 of file mqttsrc.c.

Here is the call graph for this function:

Here is the caller graph for this function:



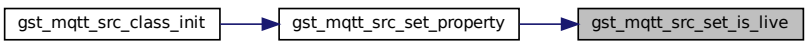
9.20.4.42 gst_mqtt_src_set_is_live()

```
static void gst_mqtt_src_set_is_live (
            GstMqttSrc * self,
            const gboolean flag )  [static]
```

Setter for the 'is-live' property.

Definition at line 892 of file mqttsrc.c.

Here is the caller graph for this function:



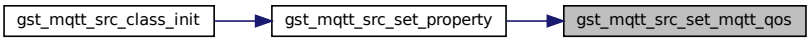
9.20.4.43 gst_mqtt_src_set_mqtt_qos()

```
static void gst_mqtt_src_set_mqtt_qos (
            GstMqttSrc * self,
            const gint qos )  [static]
```

Setter for the 'mqtt-qos' property.

Definition at line 1044 of file mqttsrc.c.

Here is the caller graph for this function:



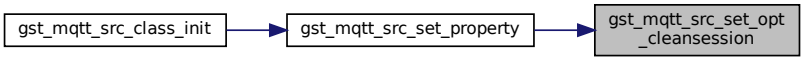
9.20.4.44 gst_mqtt_src_set_opt_cleansession()

```
static void gst_mqtt_src_set_opt_cleansession (
            GstMqttSrc * self,
            const gboolean val )  [static]
```

Setter for the 'cleansession' property.

Definition at line 1008 of file mqttsrc.c.

Here is the caller graph for this function:



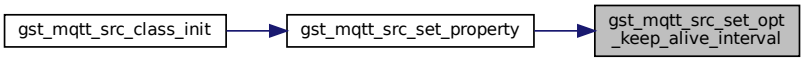
9.20.4.45 gst_mqtt_src_set_opt_keep_alive_interval()

```
static void gst_mqtt_src_set_opt_keep_alive_interval (
            GstMqttSrc * self,
            const gint num )  [static]
```

Setter for the 'keep-alive-interval' property.

Definition at line 1026 of file mqttsrc.c.

Here is the caller graph for this function:



9.20.4.46 gst_mqtt_src_set_property()

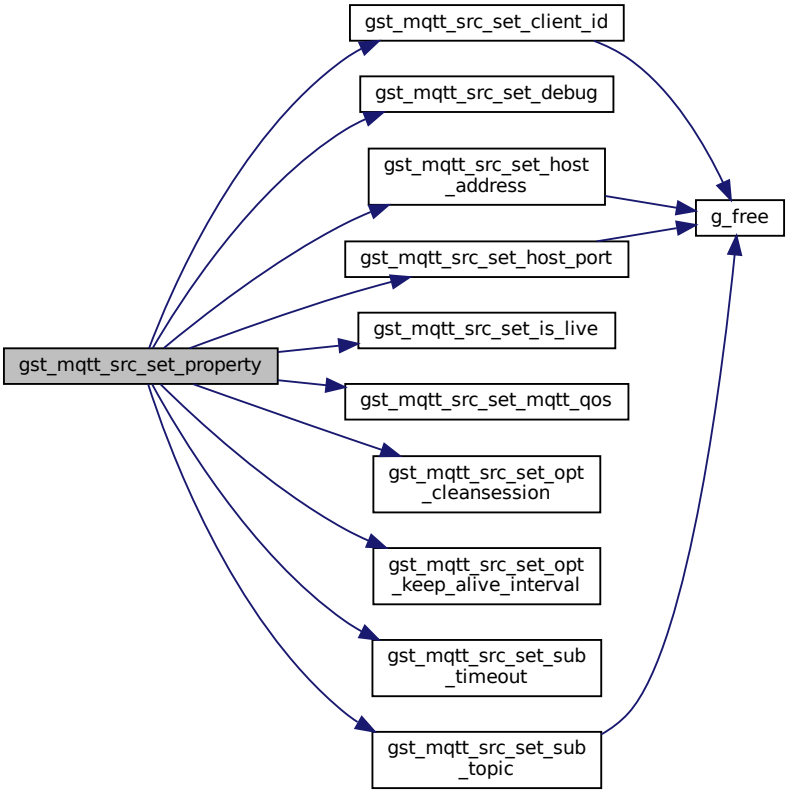
```
static void gst_mqtt_src_set_property (
            GObject * object,
            guint prop_id,
            const GValue * value,
            GParamSpec * pspec )  [static]
```

The setter for the mqttsrc's properties.

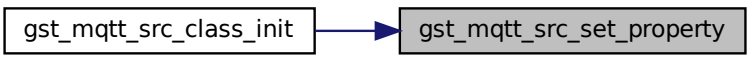
Function prototype declarations

Definition at line 316 of file mqttsrc.c.

Here is the call graph for this function:



Here is the caller graph for this function:



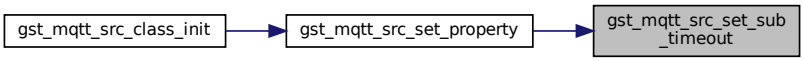
9.20.4.47 gst_mqtt_src_set_sub_timeout()

```
static void gst_mqtt_src_set_sub_timeout (
            GstMqttSrc * self,
            const gint64 t )  [static]
```

Setter for the 'sub-timeout' property.

Definition at line 971 of file mqttsrc.c.

Here is the caller graph for this function:



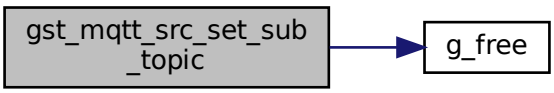
9.20.4.48 gst_mqtt_src_set_sub_topic()

```
static void gst_mqtt_src_set_sub_topic (
            GstMqttSrc * self,
            const gchar * topic )  [static]
```

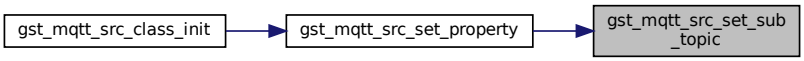
Setter for the 'sub-topic' property.

Definition at line 989 of file mqttsrc.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.20.4.49 gst_mqtt_src_start()

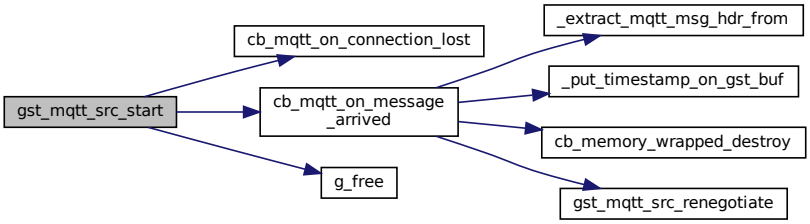
```
static gboolean gst_mqtt_src_start (
            GstBaseSrc * basesrc )  [static]
```

Start mqttsrc, called when state changed null to ready.

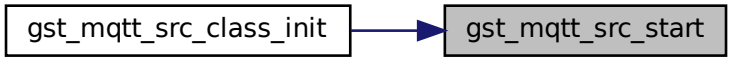
Todo Support other persistence mechanisms MQTTCLIENT_PERSISTENCE_NONE: A memory-based persistence mechanism MQTTCLIENT_PERSISTENCE_DEFAULT: The default file system-based persistence mechanism MQTTCLIENT_PERSISTENCE_USER: An application-specific persistence mechanism

Definition at line 522 of file mqttsrc.c.

Here is the call graph for this function:



Here is the caller graph for this function:



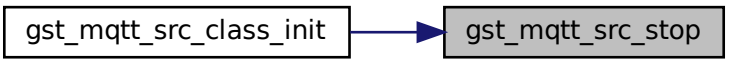
9.20.4.50 gst_mqtt_src_stop()

```
static gboolean gst_mqtt_src_stop (
            GstBaseSrc * basesrc )  [static]
```

Stop mqttsrc, called when state changed ready to null.

Definition at line 583 of file mqttsrc.c.

Here is the caller graph for this function:



9.20.5 Variable Documentation

9.20.5.1 DEFAULT_MQTT_CLIENT_ID

```
const gchar DEFAULT_MQTT_CLIENT_ID[]  [static]
```

Initial value:

```
=
    "$HOSTNAME_$PID_^[0-9][0-9]?$|^255$"
```

Definition at line 73 of file mqttsrc.c.

9.20.5.2 DEFAULT_MQTT_CLIENT_ID_FORMAT

```
const gchar DEFAULT_MQTT_CLIENT_ID_FORMAT[] = "%s_%u_src%u"  [static]
```

Definition at line 75 of file mqttsrc.c.

9.20.5.3 DEFAULT_MQTT_HOST_ADDRESS

```
const gchar DEFAULT_MQTT_HOST_ADDRESS[] = "127.0.0.1"  [static]
```

Definition at line 70 of file mqttsrc.c.

9.20.5.4 DEFAULT_MQTT_HOST_PORT

```
const gchar DEFAULT_MQTT_HOST_PORT[] = "1883"  [static]
```

Definition at line 71 of file mqttsrc.c.

9.20.5.5 src_client_id

```
guint8 src_client_id = 0  [static]
```

Definition at line 69 of file mqttsrc.c.

9.20.5.6 src_pad_template

```
GstStaticPadTemplate src_pad_template  [static]
```

Initial value:

```
= GST_STATIC_PAD_TEMPLATE ("src",
    GST_PAD_SRC, GST_PAD_ALWAYS, GST_STATIC_CAPS_ANY)
```

Definition at line 31 of file mqttsrc.c.

9.20.5.7 TAG_ERR_MQTTSRC

```
const gchar TAG_ERR_MQTTSRC[] = "ERROR: MQTTSrc"  [static]
```

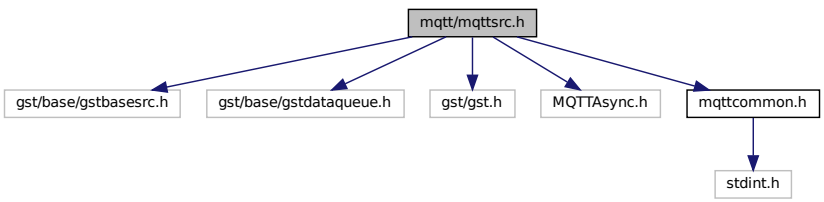
Definition at line 72 of file mqttsrc.c.

9.21 mqtt/mqttsrc.h File Reference

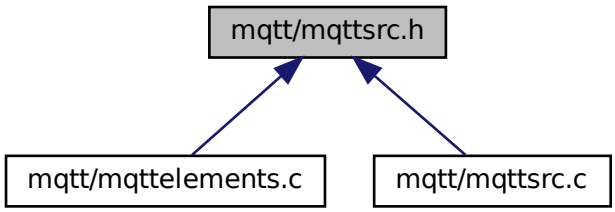
Subscribe a MQTT topic and push incoming data to the GStreamer pipeline.

```
#include <gst/base/gstbasesrc.h>
#include <gst/base/gstdataqueue.h>
#include <gst/gst.h>
#include <MQTTAsync.h>
#include "mqttcommon.h"
```

Include dependency graph for mqttsrc.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct _GstMqttSrc

  GstMqttSrc data structure.
- struct _GstMqttSrcClass

  GstMqttSrcClass data structure.

Macros

- #define GST_TYPE_MQTT_SRC (gst_mqtt_src_get_type())
- #define GST_MQTT_SRC(obj) (G_TYPE_CHECK_INSTANCE_CAST ((obj), GST_TYPE_MQTT_SRC, GstMqttSrc))
- #define GST_IS_MQTT_SRC(obj) (G_TYPE_CHECK_INSTANCE_TYPE ((obj), GST_TYPE_MQTT_SRC))
- #define GST_MQTT_SRC_CAST(obj) ((GstMqttSrc *) obj)
- #define GST_MQTT_SRC_CLASS(klass) (G_TYPE_CHECK_CLASS_CAST ((klass), GST_TYPE_MQTT_SRC, GstMqttSrcClass))
- #define GST_IS_MQTT_SRC_CLASS(klass) (G_TYPE_CHECK_CLASS_TYPE ((klass), GST_TYPE_MQTT_SRC))

Typedefs

- typedef struct _GstMqttSrc GstMqttSrc
- typedef struct _GstMqttSrcClass GstMqttSrcClass

Functions

- GType gst_mqtt_src_get_type (void)

9.21.1 Detailed Description

Subscribe a MQTT topic and push incoming data to the GStreamer pipeline.

Copyright (C) 2021 Wook Song wook16.song@samsung.com

Date

08 Mar 2021

See also

https://github.com/nnstreamer/nnstreamer

Author

Wook Song wook16.song@samsung.com

Bug No known bugs except for NYI items

9.21.2 Macro Definition Documentation

9.21.2.1 GST_IS_MQTT_SRC

```
#define GST_IS_MQTT_SRC(
              obj ) (G_TYPE_CHECK_INSTANCE_TYPE ((obj), GST_TYPE_MQTT_SRC))
```

Definition at line 29 of file mqttsrc.h.

9.21.2.2 GST_IS_MQTT_SRC_CLASS

```
#define GST_IS_MQTT_SRC_CLASS(
              klass ) (G_TYPE_CHECK_CLASS_TYPE ((klass), GST_TYPE_MQTT_SRC))
```

Definition at line 35 of file mqttsrc.h.

9.21.2.3 GST_MQTT_SRC

```
#define GST_MQTT_SRC(
              obj ) (G_TYPE_CHECK_INSTANCE_CAST ((obj), GST_TYPE_MQTT_SRC, GstMqttSrc))
```

Definition at line 27 of file mqttsrc.h.

9.21.2.4 GST_MQTT_SRC_CAST

```
#define GST_MQTT_SRC_CAST(
              obj ) ((GstMqttSrc *) obj)
```

Definition at line 31 of file mqttsrc.h.

9.21.2.5 GST_MQTT_SRC_CLASS

```
#define GST_MQTT_SRC_CLASS(
              klass ) (G_TYPE_CHECK_CLASS_CAST ((klass), GST_TYPE_MQTT_SRC, GstMqttSrcClass))
```

Definition at line 33 of file mqttsrc.h.

9.21.2.6 GST_TYPE_MQTT_SRC

```
#define GST_TYPE_MQTT_SRC (gst_mqtt_src_get_type())
```

Definition at line 25 of file mqttsrc.h.

9.21.3 Typedef Documentation

9.21.3.1 GstMqttSrc

```
typedef struct _GstMqttSrc GstMqttSrc
```

Definition at line 38 of file mqttsrc.h.

9.21.3.2 GstMqttSrcClass

```
typedef struct _GstMqttSrcClass GstMqttSrcClass
```

Definition at line 39 of file mqttsrc.h.

9.21.4 Function Documentation

9.21.4.1 gst_mqtt_src_get_type()

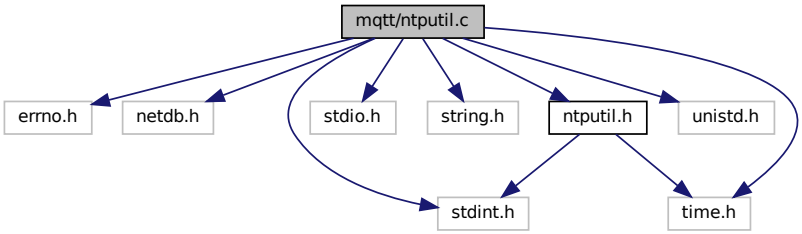
```
GType gst_mqtt_src_get_type (
            void  )
```

9.22 mqtt/ntputil.c File Reference

NTP utility functions.

```
#include <errno.h>
#include <netdb.h>
#include <stdint.h>
#include <stdio.h>
#include <string.h>
#include <time.h>
#include <unistd.h>
#include "ntputil.h"
```

Include dependency graph for ntputil.c:



Classes

- struct _ntp_timestamp_t

  A custom data type to represent NTP timestamp format.
- struct _ntp_packet_t

  A custom data type to represent NTP packet header format.

Typedefs

- typedef struct _ntp_timestamp_t ntp_timestamp_t

  A custom data type to represent NTP timestamp format.
- typedef struct _ntp_packet_t ntp_packet_t

  A custom data type to represent NTP packet header format.

Functions

- uint32_t _convert_to_host_byte_order (uint32_t in)

  Wrapper function of ntohl.
- int64_t ntputil_get_epoch (uint32_t hnums, char **hnames, uint16_t *ports)

  Get NTP timestamps from the given or public NTP servers.

Variables

- const uint64_t NTPUTIL_TIMESTAMP_DELTA = 2208988800ULL
- const double NTPUTIL_MAX_FRAC_DOUBLE = 4294967295.0L
- const int64_t NTPUTIL_SEC_TO_USEC_MULTIPLIER = 1000000
- const char NTPUTIL_DEFAULT_HNAME [ ] = "pool.ntp.org"
- const uint16_t NTPUTIL_DEFAULT_PORT = 123

9.22.1 Detailed Description

NTP utility functions.

Copyright (C) 2021 Wook Song wook16.song@samsung.com

Date

16 Jul 2021

See also

https://github.com/nnstreamer/nnstreamer

Author

Wook Song wook16.song@samsung.com

Bug No known bugs except for NYI items

Todo Need to support caching and polling timer mechanism

9.22.2 Typedef Documentation

9.22.2.1 ntp_packet_t

```
typedef struct _ntp_packet_t ntp_packet_t
```

A custom data type to represent NTP packet header format.

NTP Packet Header Format (https://www.ietf.org/rfc/rfc5905.txt p.18) 0 1 2 3 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 +-+ |LI | VN |Mode | Stratum | Poll | Precision | +-+ | Root Delay | +-+ | Root Dispersion | +-+ | Reference ID | +-+ | |

- Reference Timestamp (64) + | | +-+ | |
- Origin Timestamp (64) + | | +-+ | |
- Receive Timestamp (64) + | | +-+ | |
- Transmit Timestamp (64) + | | +-+ | | . . . Extension Field 1 (variable) . . . | | +-+ | | . . . Extension Field 2 (variable) . . . | | +-+ | Key Identifier | +-+ | | | dgst (128) | | | +-+

9.22.2.2 ntp_timestamp_t

```
typedef struct _ntp_timestamp_t ntp_timestamp_t
```

A custom data type to represent NTP timestamp format.

NTP Timestamp Format (https://www.ietf.org/rfc/rfc5905.txt p.12) 0 1 2 3 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 +-+ | Seconds | +-+ | Fraction | +-+

9.22.3 Function Documentation

9.22.3.1 _convert_to_host_byte_order()

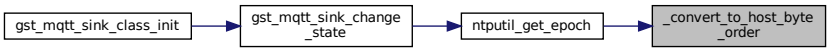
```
uint32_t _convert_to_host_byte_order (
            uint32_t in )
```

Wrapper function of ntohl.

Converting network byte order to host byte order.

Definition at line 125 of file ntputil.c.

Here is the caller graph for this function:



9.22.3.2 ntputil_get_epoch()

```
int64_t ntputil_get_epoch (
            uint32_t hnums,
            char ** hnames,
            uint16_t * ports )
```

Get NTP timestamps from the given or public NTP servers.

Parameters

in	*hnums*	A number of hostname and port pairs. If 0 is given, the NTP server pool will be used.
in	*hnames*	A list of hostname
in	*ports*	A list of port

Returns

an Unix epoch time as microseconds on success, negative values on error

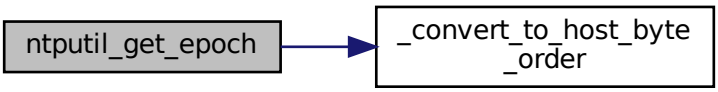
Note

ntp_timestamp_t recv_ts in ntp_packet_t means the timestamp as the packet left the NTP server. 'sec' corresponds to the seconds passed since 1900 and 'frac' is needed to convert seconds to smaller units of a second such as microsceonds. Note that the bit/byte order of those data should be converted to the host's endianness.

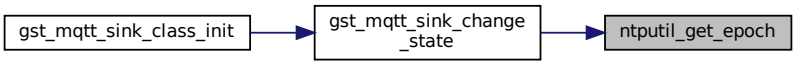
NTP uses an epoch of January 1, 1900 while the Unix epoch is the number of seconds that have elapsed since January 1, 1970. For this reason, we subtract 70 years worth of seconds from the seconds since 1900

Definition at line 140 of file ntputil.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.22.4 Variable Documentation

9.22.4.1 NTPUTIL_DEFAULT_HNAME

```
const char NTPUTIL_DEFAULT_HNAME[] = "pool.ntp.org"
```

Definition at line 118 of file ntputil.c.

9.22.4.2 NTPUTIL_DEFAULT_PORT

```
const uint16_t NTPUTIL_DEFAULT_PORT = 123
```

Definition at line 119 of file ntputil.c.

9.22.4.3 NTPUTIL_MAX_FRAC_DOUBLE

```
const double NTPUTIL_MAX_FRAC_DOUBLE = 4294967295.0L
```

Definition at line 116 of file ntputil.c.

9.22.4.4 NTPUTIL_SEC_TO_USEC_MULTIPLIER

```
const int64_t NTPUTIL_SEC_TO_USEC_MULTIPLIER = 1000000
```

Definition at line 117 of file ntputil.c.

9.22.4.5 NTPUTIL_TIMESTAMP_DELTA

```
const uint64_t NTPUTIL_TIMESTAMP_DELTA = 2208988800ULL
```

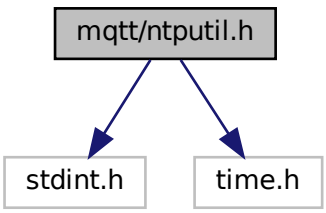
Definition at line 115 of file ntputil.c.

9.23 mqtt/ntputil.h File Reference

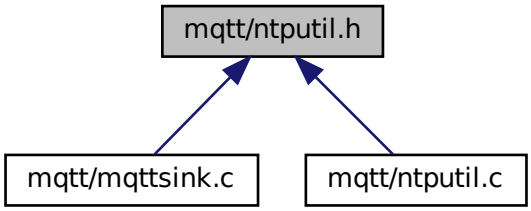
A header file of NTP utility functions.

```
#include <stdint.h>
#include <time.h>
```

Include dependency graph for ntputil.h:



This graph shows which files directly or indirectly include this file:



Functions

- int64_t ntputil_get_epoch (uint32_t hnums, char **hnames, uint16_t *ports)

  Get NTP timestamps from the given or public NTP servers.
- uint32_t _convert_to_host_byte_order (uint32_t in)

  Converting network byte order to host byte order.

9.23.1 Detailed Description

A header file of NTP utility functions.

Copyright (C) 2021 Wook Song wook16.song@samsung.com

Date

28 Jul 2021

See also

https://github.com/nnstreamer/nnstreamer

Author

Wook Song wook16.song@samsung.com

Bug No known bugs except for NYI items

9.23.2 Function Documentation

9.23.2.1 _convert_to_host_byte_order()

```
uint32_t _convert_to_host_byte_order (
            uint32_t in )
```

Converting network byte order to host byte order.

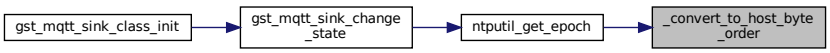
Note

There is a problem in mocking ntohl in GMock, so the wrapper function is temporarily used.

Converting network byte order to host byte order.

Definition at line 125 of file ntputil.c.

Here is the caller graph for this function:



9.23.2.2 ntputil_get_epoch()

```
int64_t ntputil_get_epoch (
            uint32_t hnums,
            char ** hnames,
            uint16_t * ports )
```

Get NTP timestamps from the given or public NTP servers.

Parameters

in	*hnums*	A number of hostname and port pairs. If 0 is given, the NTP server pool will be used.
in	*hnames*	A list of hostname
in	*ports*	A list of port

Returns

an Unix epoch time as microseconds on success, negative values on error

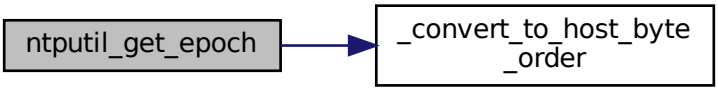
Note

ntp_timestamp_t recv_ts in ntp_packet_t means the timestamp as the packet left the NTP server. 'sec' corresponds to the seconds passed since 1900 and 'frac' is needed to convert seconds to smaller units of a second such as microsceonds. Note that the bit/byte order of those data should be converted to the host's endianness.

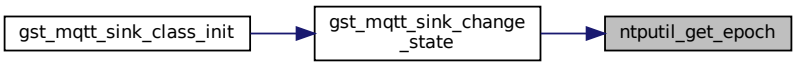
NTP uses an epoch of January 1, 1900 while the Unix epoch is the number of seconds that have elapsed since January 1, 1970. For this reason, we subtract 70 years worth of seconds from the seconds since 1900

Definition at line 140 of file ntputil.c.

Here is the call graph for this function:



Here is the caller graph for this function:

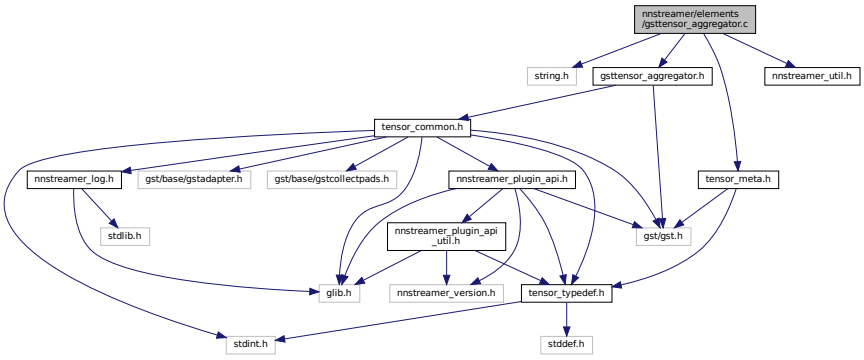


9.24 nnstreamer/elements/gsttensor_aggregator.c File Reference

GStreamer plugin to aggregate tensor stream.

```
#include <string.h>
#include "gsttensor_aggregator.h"
#include "tensor_meta.h"
#include <nnstreamer_util.h>
```

Include dependency graph for gsttensor_aggregator.c:



Macros

- #define DBG (!self->silent)

  Macro for debug mode.
- #define silent_debug_config(self, c, msg)
- #define GST_CAT_DEFAULT gst_tensor_aggregator_debug
- #define DEFAULT_SILENT TRUE

  Flag to print minimized log.
- #define DEFAULT_FRAMES_IN 1

  The number of frames in input buffer.
- #define DEFAULT_FRAMES_OUT 1

  The number of frames in output buffer.
- #define DEFAULT_FRAMES_FLUSH 0

  The number of frames to flush.
- #define DEFAULT_FRAMES_DIMENSION (NNS_TENSOR_RANK_LIMIT - 1)

  The dimension index of frames in configured tensor.
- #define DEFAULT_CONCAT TRUE

  Flag to concatenate output buffer.
- #define CAPS_STRING GST_TENSOR_CAP_DEFAULT ";" GST_TENSORS_CAP_WITH_NUM ("1")

  Template caps string for pads.
- #define gst_tensor_aggregator_parent_class parent_class

Enumerations

- enum {
  PROP_0, PROP_FRAMES_IN, PROP_FRAMES_OUT, PROP_FRAMES_FLUSH,
  PROP_FRAMES_DIMENSION, PROP_CONCAT, PROP_SILENT }

  tensor_aggregator properties

Functions

- GST_DEBUG_CATEGORY_STATIC (gst_tensor_aggregator_debug)
- G_DEFINE_TYPE (GstTensorAggregator, gst_tensor_aggregator, GST_TYPE_ELEMENT)
- static void gst_tensor_aggregator_finalize (GObject *object)

  Function to finalize instance.
- static void gst_tensor_aggregator_set_property (GObject *object, guint prop_id, const GValue *value, GParamSpec *pspec)

  Setter for tensor_aggregator properties.
- static void gst_tensor_aggregator_get_property (GObject *object, guint prop_id, GValue *value, GParamSpec *pspec)

  Getter for tensor_aggregator properties.
- static gboolean gst_tensor_aggregator_sink_event (GstPad *pad, GstObject *parent, GstEvent *event)

  This function handles sink events.
- static gboolean gst_tensor_aggregator_sink_query (GstPad *pad, GstObject *parent, GstQuery *query)

  This function handles sink pad query.
- static gboolean gst_tensor_aggregator_src_query (GstPad *pad, GstObject *parent, GstQuery *query)

  This function handles src pad query.
- static GstFlowReturn gst_tensor_aggregator_chain (GstPad *pad, GstObject *parent, GstBuffer *buf)

  Chain function, this function does the actual processing.
- static GstStateChangeReturn gst_tensor_aggregator_change_state (GstElement *element, GstStateChange transition)

Called to perform state change.

- static void gst_tensor_aggregator_reset (GstTensorAggregator *self)

  Clear and reset data.
- static GstCaps * gst_tensor_aggregator_query_caps (GstTensorAggregator *self, GstPad *pad, GstCaps *filter)

  Get pad caps for caps negotiation.
- static gboolean gst_tensor_aggregator_parse_caps (GstTensorAggregator *self, const GstCaps *caps)

  Parse caps and set tensor info.
- static void gst_tensor_aggregator_class_init (GstTensorAggregatorClass *klass)

  Initialize the tensor_aggregator's class.
- static void gst_tensor_aggregator_init (GstTensorAggregator *self)

  Initialize tensor_aggregator element.
- static GstAdapter * gst_tensor_aggregator_get_adapter (GstTensorAggregator *self, GstBuffer *buf)

  Internal function to get adapter.
- static gboolean gst_tensor_aggregator_check_concat_axis (GstTensorAggregator *self, const GstTensorInfo *info)

  Check tensor dimension and axis to concatenate data.
- static gboolean gst_tensor_aggregator_concat (GstTensorAggregator *self, GstBuffer *outbuf, const GstTensorInfo *info)

  Change the data in buffer with given axis.
- static GstFlowReturn gst_tensor_aggregator_push (GstTensorAggregator *self, GstBuffer *outbuf, gsize frame_size)

  Push the buffer to source pad. (Concatenate the buffer if needed)

Variables

- static GstStaticPadTemplate sink_template

  Template for sink pad.
- static GstStaticPadTemplate src_template

  Template for src pad.

9.24.1 Detailed Description

GStreamer plugin to aggregate tensor stream.

GStreamer Copyright (C) 2005 Thomas Vander Stichele thomas@apestaart.org Copyright (C) 2005 Ronald S. Bultje rbultje@ronald.bitfreak.net Copyright (C) 2018 Samsung Electronics Co., Ltd.

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Library General Public License as published by the Free Software Foundation; version 2.1 of the License.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Library General Public License for more details. SECTION:element-tensor_aggregator

Date

29 August 2018

See also

https://github.com/nnstreamer/nnstreamer

Author

Jaeyun Jung jy1210.jung@samsung.com

Bug No known bugs except for NYI items

9.24.2 Macro Definition Documentation

9.24.2.1 CAPS_STRING

```
#define CAPS_STRING GST_TENSOR_CAP_DEFAULT ";" GST_TENSORS_CAP_WITH_NUM ("1")
```

Template caps string for pads.

Definition at line 106 of file gsttensor_aggregator.c.

9.24.2.2 DBG

```
#define DBG (!self->silent)
```

Macro for debug mode.

Definition at line 42 of file gsttensor_aggregator.c.

9.24.2.3 DEFAULT_CONCAT

```
#define DEFAULT_CONCAT TRUE
```

Flag to concatenate output buffer.

Definition at line 101 of file gsttensor_aggregator.c.

9.24.2.4 DEFAULT_FRAMES_DIMENSION

```
#define DEFAULT_FRAMES_DIMENSION (NNS_TENSOR_RANK_LIMIT - 1)
```

The dimension index of frames in configured tensor.

Definition at line 96 of file gsttensor_aggregator.c.

9.24.2.5 DEFAULT_FRAMES_FLUSH

```
#define DEFAULT_FRAMES_FLUSH 0
```

The number of frames to flush.

Definition at line 91 of file gsttensor_aggregator.c.

9.24.2.6 DEFAULT_FRAMES_IN

```
#define DEFAULT_FRAMES_IN 1
```

The number of frames in input buffer.

Definition at line 81 of file gsttensor_aggregator.c.

9.24.2.7 DEFAULT_FRAMES_OUT

```
#define DEFAULT_FRAMES_OUT 1
```

The number of frames in output buffer.

Definition at line 86 of file gsttensor_aggregator.c.

9.24.2.8 DEFAULT_SILENT

```
#define DEFAULT_SILENT TRUE
```

Flag to print minimized log.

Definition at line 76 of file gsttensor_aggregator.c.

9.24.2.9 GST_CAT_DEFAULT

```
#define GST_CAT_DEFAULT gst_tensor_aggregator_debug
```

Definition at line 57 of file gsttensor_aggregator.c.

9.24.2.10 gst_tensor_aggregator_parent_class

```
#define gst_tensor_aggregator_parent_class parent_class
```

Definition at line 124 of file gsttensor_aggregator.c.

9.24.2.11 silent_debug_config

```
#define silent_debug_config(
              self,
              c,
              msg )
```

Value:

```
  do { \
  if (DBG) { \
    if (c) { \
      gchar *dim_str; \
      dim_str = gst_tensor_get_dimension_string ((c)->info.info[0].dimension); \
      GST_DEBUG_OBJECT (self, msg " type=%d dim=%s rate=%d/%d", (c)->info.info[0].type, dim_str,
       (c)->rate_n, (c)->rate_d); \
      g_free (dim_str); \
    } \
  } \
} while (0)
```

Definition at line 45 of file gsttensor_aggregator.c.

9.24.3 Enumeration Type Documentation

9.24.3.1 anonymous enum

```
anonymous enum
```

tensor_aggregator properties

Enumerator

PROP_0	
PROP_FRAMES_IN	
PROP_FRAMES_OUT	
PROP_FRAMES_FLUSH	
PROP_FRAMES_DIMENSION	
PROP_CONCAT	
PROP_SILENT	

Definition at line 62 of file gsttensor_aggregator.c.

9.24.4 Function Documentation

9.24.4.1 G_DEFINE_TYPE()

```
G_DEFINE_TYPE (
             GstTensorAggregator ,
```

```
            gst_tensor_aggregator ,
            GST_TYPE_ELEMENT  )
```

9.24.4.2 GST_DEBUG_CATEGORY_STATIC()

```
GST_DEBUG_CATEGORY_STATIC (
            gst_tensor_aggregator_debug  )
```

9.24.4.3 gst_tensor_aggregator_chain()

```
static GstFlowReturn gst_tensor_aggregator_chain (
            GstPad * pad,
            GstObject * parent,
            GstBuffer * buf )  [static]
```

Chain function, this function does the actual processing.

push the incoming buffer (do concat if needed)

supposed same duration for incoming buffer

Update timestamp. If frames-in is larger then frames-out, the same timestamp (pts and dts) would be returned.

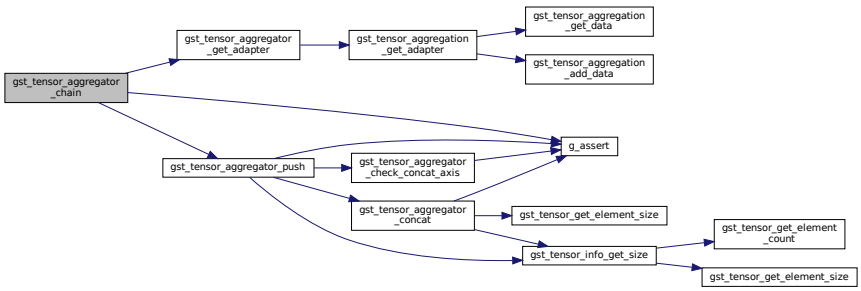
set timestamp

flush data

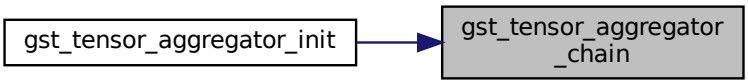
Todo flush data Invalid state, tried to flush large size. We have to determine how to handle this case. (flush the out-size or all available bytes) Now all available bytes in adapter will be flushed.

Definition at line 835 of file gsttensor_aggregator.c.

Here is the call graph for this function:



Here is the caller graph for this function:



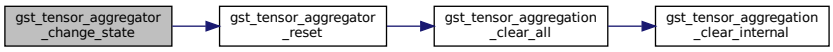
9.24.4.4 gst_tensor_aggregator_change_state()

```
static GstStateChangeReturn gst_tensor_aggregator_change_state (
            GstElement * element,
            GstStateChange transition )  [static]
```

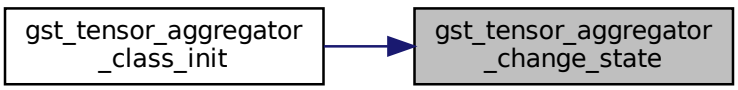
Called to perform state change.

Definition at line 947 of file gsttensor_aggregator.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.24.4.5 gst_tensor_aggregator_check_concat_axis()

```
static gboolean gst_tensor_aggregator_check_concat_axis (
            GstTensorAggregator * self,
            const GstTensorInfo * info )  [static]
```

Check tensor dimension and axis to concatenate data.

Parameters

self	this pointer to GstTensorAggregator
info	tensor info for one frame

Returns

True if needed to concatenate

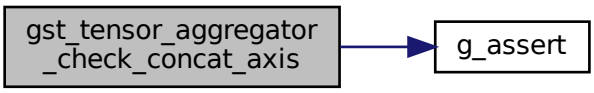
Internal error. Caller should've checked it

Check condition to concatenate data.

concatenate data

Definition at line 540 of file gsttensor_aggregator.c.

Here is the call graph for this function:



Here is the caller graph for this function:

9.24.4.6 gst_tensor_aggregator_class_init()

```
static void gst_tensor_aggregator_class_init (
            GstTensorAggregatorClass * klass )  [static]
```

Initialize the tensor_aggregator's class.

GstTensorAggregator::frames-in:

The number of frames in incoming buffer. GstTensorAggregator itself cannot get the frames in buffer. (buffer is a sinle tensor instance) GstTensorAggregator calculates the size of single frame with this property.

GstTensorAggregator::frames-out:

The number of frames in outgoing buffer. (buffer is a sinle tensor instance) GstTensorAggregator calculates the size of outgoing frames and pushes a buffer to source pad.

GstTensorAggregator::frames-flush:

The number of frames to flush. GstTensorAggregator flushes the bytes (N frames) in GstAdapter after pushing a buffer. If set 0 (default value), all outgoing frames will be flushed.

GstTensorAggregator::frames-dim:

The dimension index of frames in tensor. If frames-in and frames-out are different, GstTensorAggregator has to change the dimension of tensor. With this property, GstTensorAggregator changes the out-caps.

GstTensorAggregator::concat:

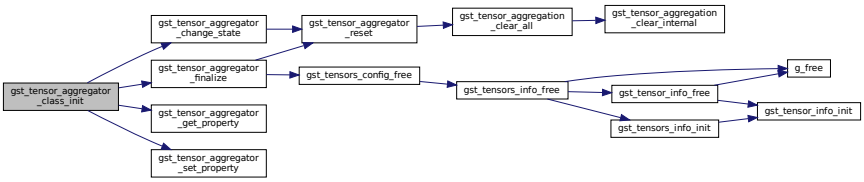
The flag to concatenate output buffer. If concat is true and frames-out is larger than 1, GstTensorAggregator will concatenate the output buffer with the axis frames-dim.

GstTensorAggregator::silent:

The flag to enable/disable debugging messages.

Definition at line 155 of file gsttensor_aggregator.c.

Here is the call graph for this function:



9.24.4.7 gst_tensor_aggregator_concat()

```
static gboolean gst_tensor_aggregator_concat (
            GstTensorAggregator * self,
            GstBuffer * outbuf,
            const GstTensorInfo * info )  [static]
```

Change the data in buffer with given axis.

Parameters

self	this pointer to GstTensorAggregator
outbuf	buffer to be concatenated
info	tensor info for one frame

Internal error

Concatenate output buffer with given axis (frames-dim) If frames-dim is equal to (NNS_TENSOR_RANK_LIMIT - 1), nothing to do. (In this case, this function will not be called. See gst_tensor_aggregator_check_concat_axis ())

Ex1) concatenate 2 frames with dimension 3:4:2:1

frame 1 [[[1101 1102 1103] [1104 1105 1106] [1107 1108 1109] [1110 1111 1112]] [[1113 1114 1115] [1116 1117 1118] [1119 1120 1121] [1122 1123 1124]]]

frame 2 [[[2101 2102 2103] [2104 2105 2106] [2107 2108 2109] [2110 2111 2112]] [[2113 2114 2115] [2116 2117 2118] [2119 2120 2121] [2122 2123 2124]]]

1-1. result with frames-dim 3 (3:4:2:2) [[[1101 1102 1103] [1104 1105 1106] [1107 1108 1109] [1110 1111 1112]] [[1113 1114 1115] [1116 1117 1118] [1119 1120 1121] [1122 1123 1124]]] [[[2101 2102 2103] [2104 2105 2106] [2107 2108 2109] [2110 2111 2112]] [[2113 2114 2115] [2116 2117 2118] [2119 2120 2121] [2122 2123 2124]]]

1-2. result with frames-dim 2 (3:4:4:1) [[[1101 1102 1103] [1104 1105 1106] [1107 1108 1109] [1110 1111 1112]] [[1113 1114 1115] [1116 1117 1118] [1119 1120 1121] [1122 1123 1124]] [[2101 2102 2103] [2104 2105 2106] [2107 2108 2109] [2110 2111 2112]] [[2113 2114 2115] [2116 2117 2118] [2119 2120 2121] [2122 2123 2124]]]

1-3. result with frames-dim 1 (3:8:2:1) [[[1101 1102 1103] [1104 1105 1106] [1107 1108 1109] [1110 1111 1112] [2101 2102 2103] [2104 2105 2106] [2107 2108 2109] [2110 2111 2112]] [[1113 1114 1115] [1116 1117 1118] [1119 1120 1121] [1122 1123 1124] [2113 2114 2115] [2116 2117 2118] [2119 2120 2121] [2122 2123 2124]]]

1-4. result with frames-dim 0 (6:4:2:1) [[[1101 1102 1103 2101 2102 2103] [1104 1105 1106 2104 2105 2106] [1107 1108 1109 2107 2108 2109] [1110 1111 1112 2110 2111 2112]] [[1113 1114 1115 2113 2114 2115] [1116 1117 1118 2116 2117 2118] [1119 1120 1121 2119 2120 2121] [1122 1123 1124 2122 2123 2124]]]

Ex2) concatenate 2 frames with dimension 3:4:2:2

frame 1 [[[1101 1102 1103] [1104 1105 1106] [1107 1108 1109] [1110 1111 1112]] [[1113 1114 1115] [1116 1117 1118] [1119 1120 1121] [1122 1123 1124]]] [[[1201 1202 1203] [1204 1205 1206] [1207 1208 1209] [1210 1211 1212]] [[1213 1214 1215] [1216 1217 1218] [1219 1220 1221] [1222 1223 1224]]]

frame 2 [[[2101 2102 2103] [2104 2105 2106] [2107 2108 2109] [2110 2111 2112]] [[2113 2114 2115] [2116 2117 2118] [2119 2120 2121] [2122 2123 2124]]] [[[2201 2202 2203] [2204 2205 2206] [2207 2208 2209] [2210 2211 2212]] [[2213 2214 2215] [2216 2217 2218] [2219 2220 2221] [2222 2223 2224]]]

2-1. result with frames-dim 3 (3:4:2:4) [[[1101 1102 1103] [1104 1105 1106] [1107 1108 1109] [1110 1111 1112]] [[1113 1114 1115] [1116 1117 1118] [1119 1120 1121] [1122 1123 1124]]] [[[1201 1202 1203] [1204 1205 1206] [1207 1208 1209] [1210 1211 1212]] [[1213 1214 1215] [1216 1217 1218] [1219 1220 1221] [1222 1223 1224]]] [[[2101 2102 2103] [2104 2105 2106] [2107 2108 2109] [2110 2111 2112]] [[2113 2114 2115] [2116 2117 2118] [2119 2120 2121] [2122 2123 2124]]] [[[2201 2202 2203] [2204 2205 2206] [2207 2208 2209] [2210 2211 2212]] [[2213 2214 2215] [2216 2217 2218] [2219 2220 2221] [2222 2223 2224]]]

2-2. result with frames-dim 2 (3:4:4:2) [[[1101 1102 1103] [1104 1105 1106] [1107 1108 1109] [1110 1111 1112]] [[1113 1114 1115] [1116 1117 1118] [1119 1120 1121] [1122 1123 1124]] [[2101 2102 2103] [2104 2105 2106] [2107 2108 2109] [2110 2111 2112]] [[2113 2114 2115] [2116 2117 2118] [2119 2120 2121] [2122 2123 2124]]] [[[1201 1202 1203] [1204 1205 1206] [1207 1208 1209] [1210 1211 1212]] [[1213 1214 1215] [1216 1217 1218] [1219 1220 1221] [1222 1223 1224]] [[2201 2202 2203] [2204 2205 2206] [2207 2208 2209] [2210 2211 2212]] [[2213 2214 2215] [2216 2217 2218] [2219 2220 2221] [2222 2223 2224]]]

2-3. result with frames-dim 1 (3:8:2:2) [[[1101 1102 1103] [1104 1105 1106] [1107 1108 1109] [1110 1111 1112] [2101 2102 2103] [2104 2105 2106] [2107 2108 2109] [2110 2111 2112]] [[1113 1114 1115] [1116 1117 1118] [1119 1120 1121] [1122 1123 1124] [2113 2114 2115] [2116 2117 2118] [2119 2120 2121] [2122 2123 2124]]] [[[1201 1202 1203] [1204 1205 1206] [1207 1208 1209] [1210 1211 1212] [2201 2202 2203] [2204 2205 2206] [2207 2208 2209] [2210 2211 2212]] [[1213 1214 1215] [1216 1217 1218] [1219 1220 1221] [1222 1223 1224] [2213 2214 2215] [2216 2217 2218] [2219 2220 2221] [2222 2223 2224]]]

2-4. result with frames-dim 0 (6:4:2:2) [[[1101 1102 1103 2101 2102 2103] [1104 1105 1106 2104 2105 2106] [1107 1108 1109 2107 2108 2109] [1110 1111 1112 2110 2111 2112]] [[1113 1114 1115 2113 2114 2115] [1116 1117 1118 2116 2117 2118] [1119 1120 1121 2119 2120 2121] [1122 1123 1124 2122 2123 2124]]] [[[1201 1202 1203 2201 2202 2203] [1204 1205 1206 2204 2205 2206] [1207 1208 1209 2207 2208 2209] [1210 1211 1212 2210 2211 2212]] [[1213 1214 1215 2213 2214 2215] [1216 1217 1218 2216 2217 2218] [1219 1220 1221 2219 2220 2221] [1222 1223 1224 2222 2223 2224]]]

Ex3) concatenate 2 frames with dimension 3:4:1:1

frame 1 [[[1101 1102 1103] [1104 1105 1106] [1107 1108 1109] [1110 1111 1112]]]

frame 2 [[[2101 2102 2103] [2104 2105 2106] [2107 2108 2109] [2110 2111 2112]]]

3-1. result with frames-dim 3 (3:4:1:2) [[[1101 1102 1103] [1104 1105 1106] [1107 1108 1109] [1110 1111 1112]]] [[[2101 2102 2103] [2104 2105 2106] [2107 2108 2109] [2110 2111 2112]]]

3-2. result with frames-dim 2 (3:4:2:1) [[[1101 1102 1103] [1104 1105 1106] [1107 1108 1109] [1110 1111 1112]] [[2101 2102 2103] [2104 2105 2106] [2107 2108 2109] [2110 2111 2112]]]

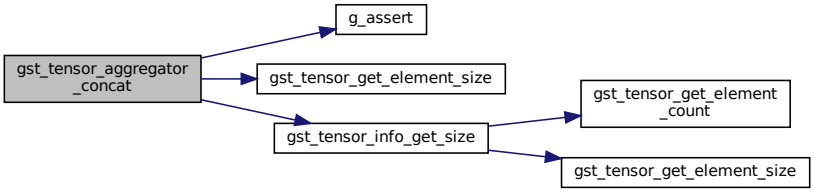
3-3. result with frames-dim 1 (3:8:1:1) [[[1101 1102 1103] [1104 1105 1106] [1107 1108 1109] [1110 1111 1112] [2101 2102 2103] [2104 2105 2106] [2107 2108 2109] [2110 2111 2112]]]

3-4. result with frames-dim 0 (6:4:1:1) [[[1101 1102 1103 2101 2102 2103] [1104 1105 1106 2104 2105 2106] [1107 1108 1109 2107 2108 2109] [1110 1111 1112 2110 2111 2112]]]

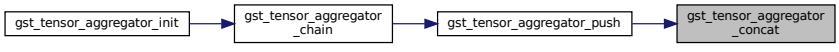
get block size

Definition at line 569 of file gsttensor_aggregator.c.

Here is the call graph for this function:



Here is the caller graph for this function:



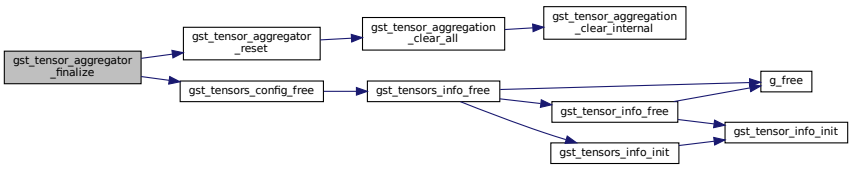
9.24.4.8 gst_tensor_aggregator_finalize()

```
static void gst_tensor_aggregator_finalize (
            GObject * object )  [static]
```

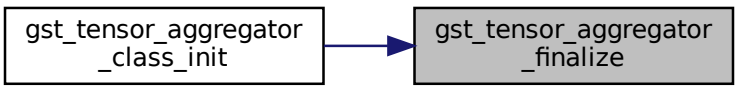
Function to finalize instance.

Definition at line 294 of file gsttensor_aggregator.c.

Here is the call graph for this function:



Here is the caller graph for this function:



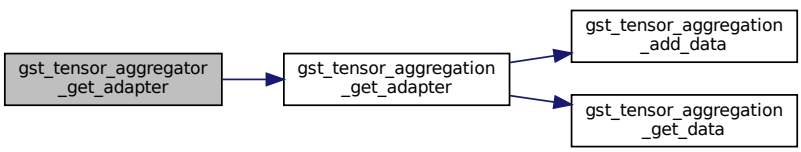
9.24.4.9 gst_tensor_aggregator_get_adapter()

```
static GstAdapter* gst_tensor_aggregator_get_adapter (
            GstTensorAggregator * self,
            GstBuffer * buf )  [static]
```

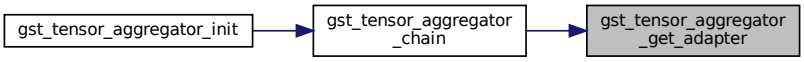
Internal function to get adapter.

Definition at line 521 of file gsttensor_aggregator.c.

Here is the call graph for this function:



Here is the caller graph for this function:



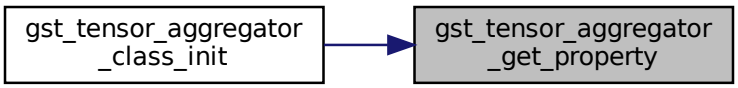
9.24.4.10 gst_tensor_aggregator_get_property()

```
static void gst_tensor_aggregator_get_property (
            GObject * object,
            guint prop_id,
            GValue * value,
            GParamSpec * pspec )  [static]
```

Getter for tensor_aggregator properties.

Definition at line 349 of file gsttensor_aggregator.c.

Here is the caller graph for this function:



9.24.4.11 gst_tensor_aggregator_init()

```
static void gst_tensor_aggregator_init (
            GstTensorAggregator * self )  [static]
```

Initialize tensor_aggregator element.

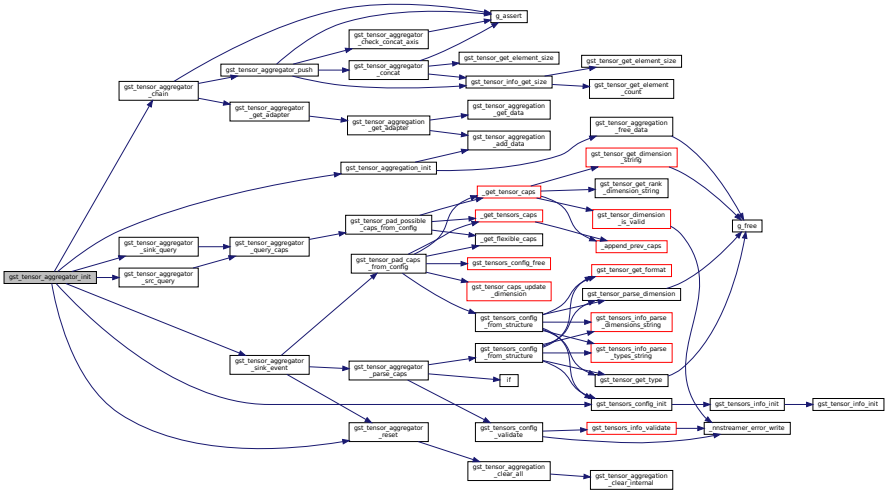
setup sink pad

setup src pad

init properties

Definition at line 254 of file gsttensor_aggregator.c.

Here is the call graph for this function:



9.24.4.12 gst_tensor_aggregator_parse_caps()

```
static gboolean gst_tensor_aggregator_parse_caps (
            GstTensorAggregator * self,
            const GstCaps * caps )  [static]
```

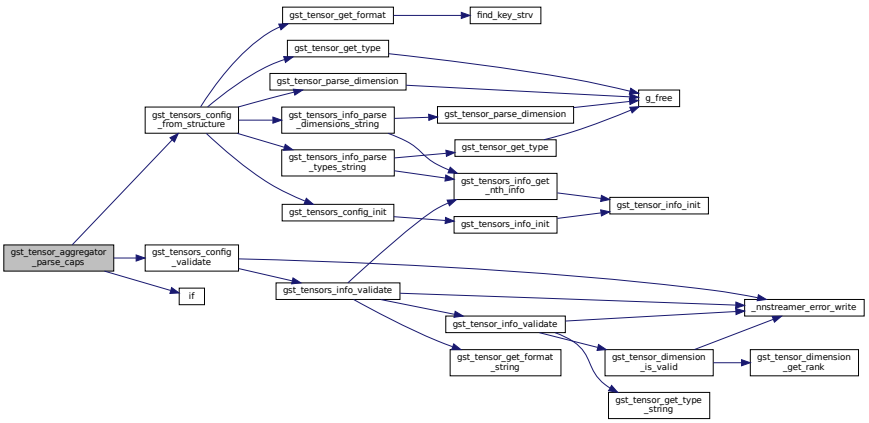
Parse caps and set tensor info.

Todo flush data Check properties to detect invalid case. Assertion when in=5 out=10 flush=20 or in=10 out=5 flush=20

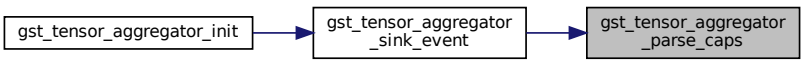
update dimension in output tensor. e.g, in-dimension 2:200:200:1 if frames_out=10 and frames_dim=3, then out-dimension is 2:200:200:10. if frames_out=10 and frames_dim=2, then out-dimension is 2:200:2000:1.

Definition at line 1026 of file gsttensor_aggregator.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.24.4.13 gst_tensor_aggregator_push()

```
static GstFlowReturn gst_tensor_aggregator_push (
            GstTensorAggregator * self,
            GstBuffer * outbuf,
            gsize frame_size )  [static]
```

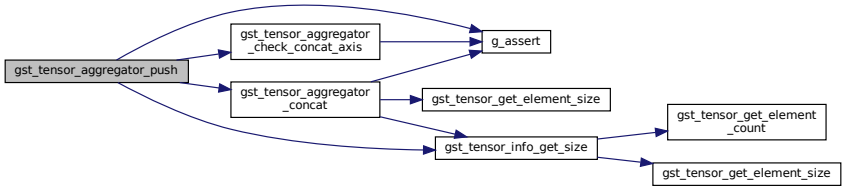
Push the buffer to source pad. (Concatenate the buffer if needed)

tensor info for one frame

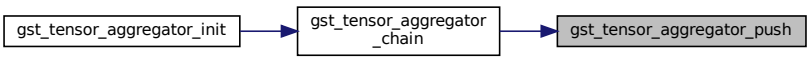
change data in buffer with given axis

Definition at line 805 of file gsttensor_aggregator.c.

Here is the call graph for this function:



Here is the caller graph for this function:



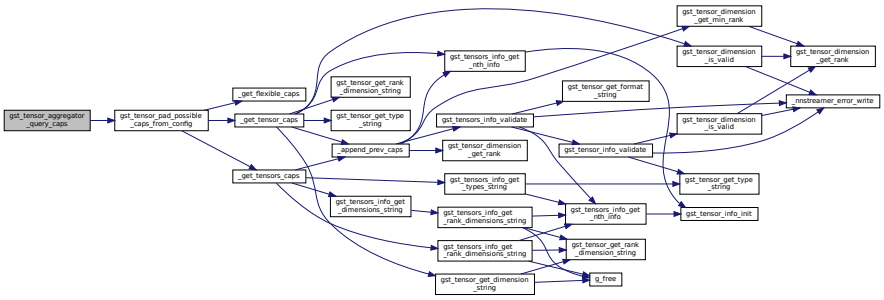
9.24.4.14 gst_tensor_aggregator_query_caps()

```
static GstCaps * gst_tensor_aggregator_query_caps (
            GstTensorAggregator * self,
            GstPad * pad,
            GstCaps * filter )  [static]
```

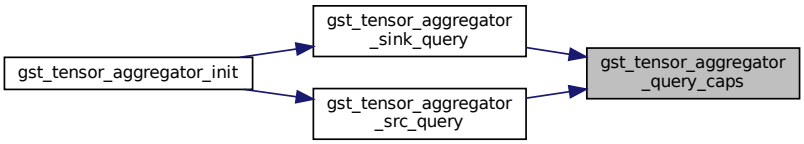
Get pad caps for caps negotiation.

Definition at line 990 of file gsttensor_aggregator.c.

Here is the call graph for this function:



Here is the caller graph for this function:



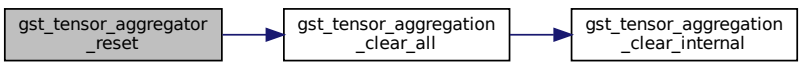
9.24.4.15 gst_tensor_aggregator_reset()

```
static void gst_tensor_aggregator_reset (
            GstTensorAggregator * self )  [static]
```

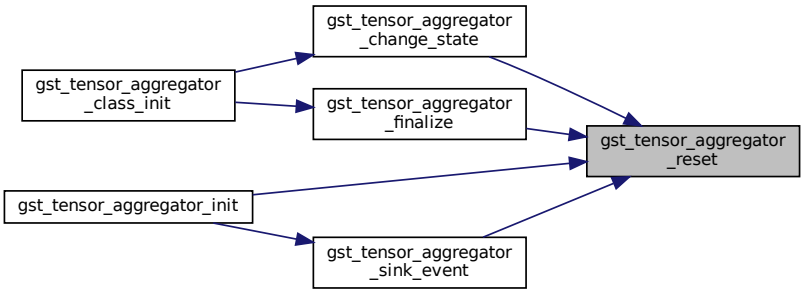
Clear and reset data.

Definition at line 980 of file gsttensor_aggregator.c.

Here is the call graph for this function:



Here is the caller graph for this function:



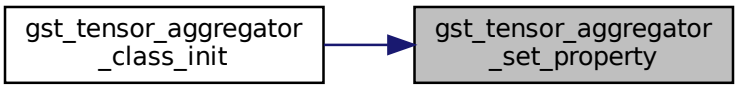
9.24.4.16 gst_tensor_aggregator_set_property()

```
static void gst_tensor_aggregator_set_property (
            GObject * object,
            guint prop_id,
            const GValue * value,
            GParamSpec * pspec )  [static]
```

Setter for tensor_aggregator properties.

Definition at line 313 of file gsttensor_aggregator.c.

Here is the caller graph for this function:



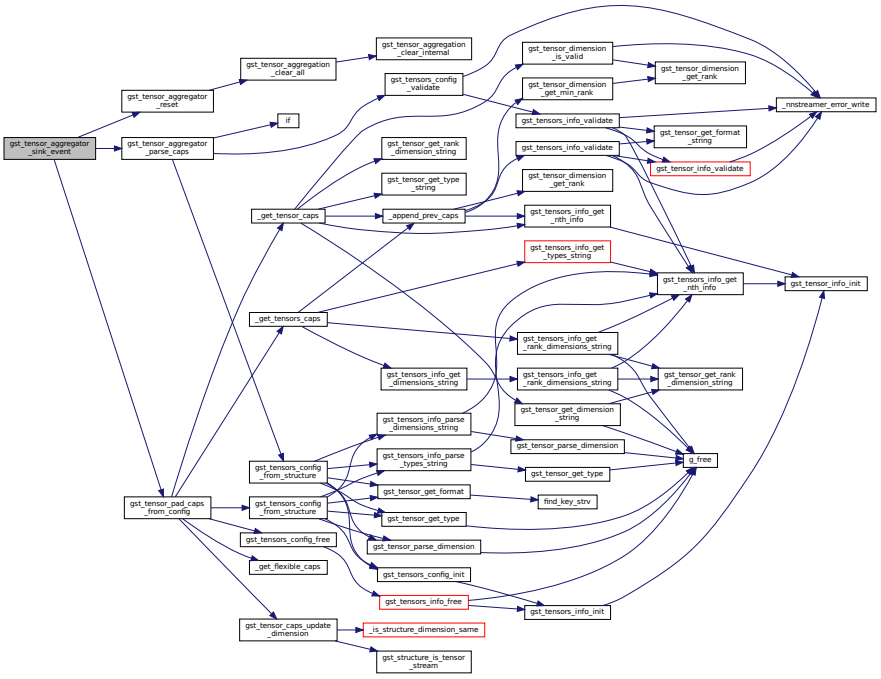
9.24.4.17 gst_tensor_aggregator_sink_event()

```
static gboolean gst_tensor_aggregator_sink_event (
            GstPad * pad,
            GstObject * parent,
            GstEvent * event )  [static]
```

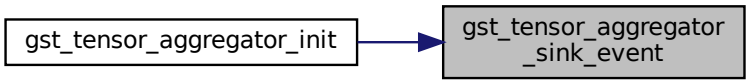
This function handles sink events.

Definition at line 385 of file gsttensor_aggregator.c.

Here is the call graph for this function:



Here is the caller graph for this function:



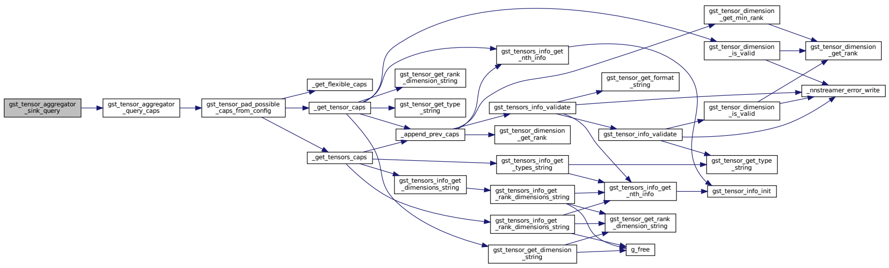
9.24.4.18 gst_tensor_aggregator_sink_query()

```
static gboolean gst_tensor_aggregator_sink_query (
            GstPad * pad,
            GstObject * parent,
            GstQuery * query )  [static]
```

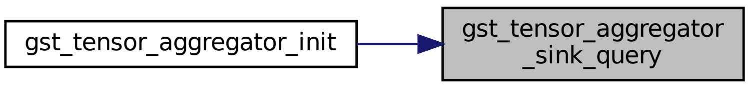
This function handles sink pad query.

Definition at line 434 of file gsttensor_aggregator.c.

Here is the call graph for this function:



Here is the caller graph for this function:



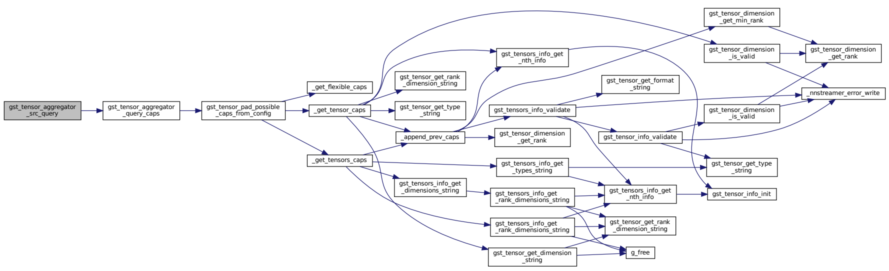
9.24.4.19 gst_tensor_aggregator_src_query()

```
static gboolean gst_tensor_aggregator_src_query (
            GstPad * pad,
            GstObject * parent,
            GstQuery * query )  [static]
```

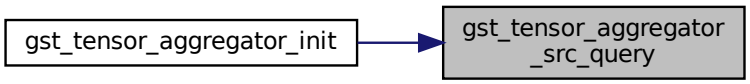
This function handles src pad query.

Definition at line 487 of file gsttensor_aggregator.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.24.5 Variable Documentation

9.24.5.1 sink_template

```
GstStaticPadTemplate sink_template  [static]
```

Initial value:

```
= GST_STATIC_PAD_TEMPLATE ("sink",
    GST_PAD_SINK,
    GST_PAD_ALWAYS,
    GST_STATIC_CAPS (CAPS_STRING))
```

Template for sink pad.

Definition at line 111 of file gsttensor_aggregator.c.

9.24.5.2 src_template

```
GstStaticPadTemplate src_template  [static]
```

Initial value:

```
= GST_STATIC_PAD_TEMPLATE ("src",
    GST_PAD_SRC,
    GST_PAD_ALWAYS,
    GST_STATIC_CAPS (CAPS_STRING))
```

Template for src pad.

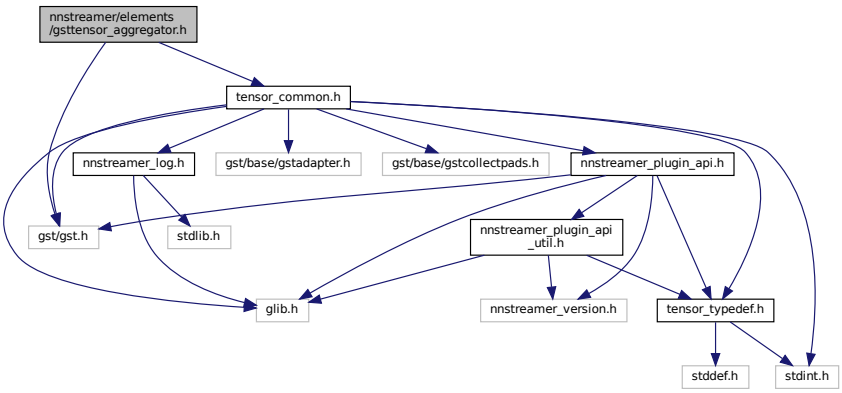
Definition at line 119 of file gsttensor_aggregator.c.

9.25 nnstreamer/elements/gsttensor_aggregator.h File Reference

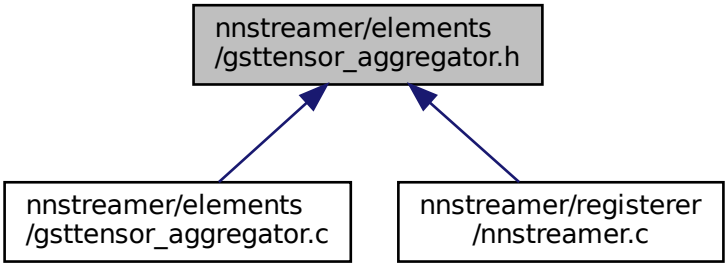
GStreamer plugin to aggregate tensor stream.

```
#include <gst/gst.h>
#include <tensor_common.h>
```

Include dependency graph for gsttensor_aggregator.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct _GstTensorAggregator

  GstTensorAggregator data structure.
- struct _GstTensorAggregatorClass

  GstTensorAggregatorClass data structure.

Macros

- #define GST_TYPE_TENSOR_AGGREGATOR (gst_tensor_aggregator_get_type())
- #define GST_TENSOR_AGGREGATOR(obj) (G_TYPE_CHECK_INSTANCE_CAST((obj),GST_TYPE_TENSOR_AGGREGA
- #define GST_TENSOR_AGGREGATOR_CLASS(klass) (G_TYPE_CHECK_CLASS_CAST((klass),GST_TYPE_TENSOR_AG
- #define GST_IS_TENSOR_AGGREGATOR(obj) (G_TYPE_CHECK_INSTANCE_TYPE((obj),GST_TYPE_TENSOR_AGGRE
- #define GST_IS_TENSOR_AGGREGATOR_CLASS(klass) (G_TYPE_CHECK_CLASS_TYPE((klass),GST_TYPE_TENSOR_

Typedefs

- typedef struct _GstTensorAggregator GstTensorAggregator
- typedef struct _GstTensorAggregatorClass GstTensorAggregatorClass

Functions

- GType gst_tensor_aggregator_get_type (void)

  Function to get type of tensor_aggregator.

9.25.1 Detailed Description

GStreamer plugin to aggregate tensor stream.

GStreamer Copyright (C) 2005 Thomas Vander Stichele thomas@apestaart.org Copyright (C) 2005 Ronald S. Bultje rbultje@ronald.bitfreak.net Copyright (C) 2018 Samsung Electronics Co., Ltd.

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Library General Public License as published by the Free Software Foundation; version 2.1 of the License.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Library General Public License for more details.

Date

29 August 2018

See also

https://github.com/nnstreamer/nnstreamer

Author

Jaeyun Jung jy1210.jung@samsung.com

Bug No known bugs except for NYI items

9.25.2 Macro Definition Documentation

9.25.2.1 GST_IS_TENSOR_AGGREGATOR

```
#define GST_IS_TENSOR_AGGREGATOR(
              obj ) (G_TYPE_CHECK_INSTANCE_TYPE((obj),GST_TYPE_TENSOR_AGGREGATOR))
```

Definition at line 41 of file gsttensor_aggregator.h.

9.25.2.2 GST_IS_TENSOR_AGGREGATOR_CLASS

```
#define GST_IS_TENSOR_AGGREGATOR_CLASS(
              klass ) (G_TYPE_CHECK_CLASS_TYPE((klass),GST_TYPE_TENSOR_AGGREGATOR))
```

Definition at line 43 of file gsttensor_aggregator.h.

9.25.2.3 GST_TENSOR_AGGREGATOR

```
#define GST_TENSOR_AGGREGATOR(
              obj ) (G_TYPE_CHECK_INSTANCE_CAST((obj),GST_TYPE_TENSOR_AGGREGATOR,GstTensorAggregator))
```

Definition at line 37 of file gsttensor_aggregator.h.

9.25.2.4 GST_TENSOR_AGGREGATOR_CLASS

```
#define GST_TENSOR_AGGREGATOR_CLASS(
              klass ) (G_TYPE_CHECK_CLASS_CAST((klass),GST_TYPE_TENSOR_AGGREGATOR,GstTensorAggregatorClass))
```

Definition at line 39 of file gsttensor_aggregator.h.

9.25.2.5 GST_TYPE_TENSOR_AGGREGATOR

```
#define GST_TYPE_TENSOR_AGGREGATOR (gst_tensor_aggregator_get_type())
```

Definition at line 35 of file gsttensor_aggregator.h.

9.25.3 Typedef Documentation

9.25.3.1 GstTensorAggregator

```
typedef struct _GstTensorAggregator GstTensorAggregator
```

Definition at line 46 of file gsttensor_aggregator.h.

9.25.3.2 GstTensorAggregatorClass

```
typedef struct _GstTensorAggregatorClass GstTensorAggregatorClass
```

Definition at line 47 of file gsttensor_aggregator.h.

9.25.4 Function Documentation

9.25.4.1 gst_tensor_aggregator_get_type()

```
GType gst_tensor_aggregator_get_type (
            void  )
```

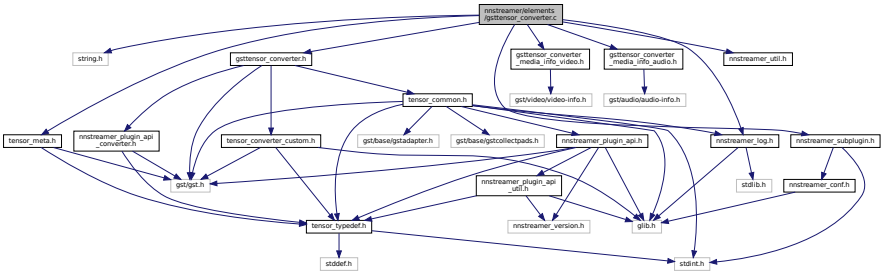
Function to get type of tensor_aggregator.

9.26 nnstreamer/elements/gsttensor_converter.c File Reference

GStreamer plugin to convert media types to tensors (as a filter for other general neural network filters)

```
#include <string.h>
#include "gsttensor_converter.h"
#include "tensor_meta.h"
#include "gsttensor_converter_media_info_video.h"
#include "gsttensor_converter_media_info_audio.h"
#include <nnstreamer_log.h>
#include <nnstreamer_subplugin.h>
#include <nnstreamer_util.h>
```

Include dependency graph for gsttensor_converter.c:



Macros

- #define TEXT_CAPS_STR "text/x-raw, format = (string) utf8"

  Caps string for text input.
- #define append_text_caps_template(caps) gst_caps_append (caps, gst_caps_from_string (TEXT_CAPS_STR))
- #define OCTET_CAPS_STR "application/octet-stream"

  Caps string for binary stream.
- #define append_octet_caps_template(caps) gst_caps_append (caps, gst_caps_from_string (OCTET_CAPS_STR))
- #define append_flex_tensor_caps_template(caps) gst_caps_append (caps, gst_caps_from_string (GST_TENSORS_FLEX_CAP_DEFAULT))

  Macro to append template caps for flexible tensor.
- #define DBG (!self->silent)

  Macro for debug mode.
- #define silent_debug_timestamp(self, buf)
- #define GST_CAT_DEFAULT gst_tensor_converter_debug
- #define STRING_CUSTOM_MODE(self)
- #define DEFAULT_SET_TIMESTAMP TRUE

  Flag to set timestamp when received a buffer with invalid timestamp.
- #define DEFAULT_SILENT TRUE

  Flag to print minimized log.
- #define DEFAULT_FRAMES_PER_TENSOR 1

  Frames in output tensor.
- #define gst_tensor_converter_parent_class parent_class

Enumerations

- enum {
  PROP_0, PROP_INPUT_DIMENSION, PROP_INPUT_TYPE, PROP_FRAMES_PER_TENSOR,
  PROP_SET_TIMESTAMP, PROP_SUBPLUGINS, PROP_SILENT, PROP_MODE }

  tensor_converter properties

Functions

- GST_DEBUG_CATEGORY_STATIC (gst_tensor_converter_debug)
- G_DEFINE_TYPE (GstTensorConverter, gst_tensor_converter, GST_TYPE_ELEMENT)
- static void gst_tensor_converter_finalize (GObject *object)

  Function to finalize instance.
- static void gst_tensor_converter_set_property (GObject *object, guint prop_id, const GValue *value, GParamSpec *pspec)

  Setter for tensor_converter properties.
- static void gst_tensor_converter_get_property (GObject *object, guint prop_id, GValue *value, GParamSpec *pspec)

  Getter for tensor_converter properties.
- static gboolean gst_tensor_converter_sink_event (GstPad *pad, GstObject *parent, GstEvent *event)

  This function handles sink event.
- static gboolean gst_tensor_converter_sink_query (GstPad *pad, GstObject *parent, GstQuery *query)

  This function handles sink pad query.
- static gboolean gst_tensor_converter_src_query (GstPad *pad, GstObject *parent, GstQuery *query)

  This function handles src pad query.
- static GstFlowReturn gst_tensor_converter_chain (GstPad *pad, GstObject *parent, GstBuffer *buf)

  Chain function, this function does the actual processing.

- static GstStateChangeReturn gst_tensor_converter_change_state (GstElement *element, GstStateChange transition)

  Called to perform state change.
- static void gst_tensor_converter_reset (GstTensorConverter *self)

  Clear and reset data.
- static GstCaps * gst_tensor_converter_query_caps (GstTensorConverter *self, GstPad *pad, GstCaps *filter)

  Get pad caps for caps negotiation.
- static gboolean gst_tensor_converter_parse_caps (GstTensorConverter *self, const GstCaps *caps)

  Parse caps and set tensors info.
- static void gst_tensor_converter_update_caps (GstTensorConverter *self)

  Update src pad caps from tensors config.
- static const NNStreamerExternalConverter * findExternalConverter (const char *media_type)

  Internal static function to find registered subplugins.
- static void gst_tensor_converter_class_init (GstTensorConverterClass *klass)

  Initialize the tensor_converter's class.
- static void gst_tensor_converter_init (GstTensorConverter *self)

  Initialize tensor_converter element.
- static GstAdapter * gst_tensor_converter_get_adapter (GstTensorConverter *self, GstBuffer *buf)

  Internal function to get adapter.
- static void _gst_tensor_converter_chain_segment (GstTensorConverter *self, gsize frame_size)

  Chain function's private routine.
- static void _gst_tensor_converter_chain_timestamp (GstTensorConverter *self, GstBuffer *inbuf, guint frames_in)

  Chain function's private routine.
- static GstBuffer * _gst_tensor_converter_chain_octet (GstTensorConverter *self, GstBuffer *buf)

  Chain function's private routine to process octet stream.
- static GstBuffer * _gst_tensor_converter_chain_flex_tensor (GstTensorConverter *self, GstBuffer *buf)

  Chain function's private routine to process flex tensor.
- static GstFlowReturn _gst_tensor_converter_chain_push (GstTensorConverter *self, GstBuffer *buf)

  Chain function's private routine to push buffer into src pad.
- static GstFlowReturn _gst_tensor_converter_chain_chunk (GstTensorConverter *self, GstBuffer *inbuf, guint frames_in, guint frames_out, gsize frame_size)

  Chain function's private routine to push multiple buffers.
- static void gst_tensor_converter_get_format_list (GValue *list,...)

  Get supported format list.
- static gboolean gst_tensor_converter_video_stride (GstVideoFormat format, gint width)

  Determine if we need zero-padding.
- static gboolean gst_tensor_converter_parse_video (GstTensorConverter *self, const GstCaps *caps, GstTensorsConfig *config)

  Set the tensors config structure from video info (internal static function)
- static gboolean gst_tensor_converter_parse_audio (GstTensorConverter *self, const GstCaps *caps, GstTensorsConfig *config)

  Set the tensors config structure from audio info (internal static function)
- static gboolean gst_tensor_converter_parse_text (GstTensorConverter *self, GstTensorsConfig *config, const GstStructure *structure)

  Set the tensors config structure from text info (internal static function)
- static gboolean gst_tensor_converter_parse_octet (GstTensorConverter *self, GstTensorsConfig *config, const GstStructure *structure)

  Set the tensors configs structure from octet stream (internal static function)
- static gboolean gst_tensor_converter_parse_tensor (GstTensorConverter *self, GstTensorsConfig *config, const GstStructure *structure)

Set the tensors configs structure from fliex tensor stream (internal static function)

- static gboolean gst_tensor_converter_parse_custom (GstTensorConverter *self, GstTensorsConfig *config, const GstCaps *caps)

  Set the tensors config structure from caps (internal static function for custom mode)
- static GstCaps * gst_tensor_converter_get_possible_media_caps (GstTensorConverter *self)

  Get possible media-caps from downstream element.
- const NNStreamerExternalConverter * nnstreamer_converter_find (const char *name)

  Find converter sub-plugin with the name.
- static gboolean nnstreamer_converter_validate (const NNStreamerExternalConverter *converter)

  Validate converter sub-plugin's data.
- int registerExternalConverter (NNStreamerExternalConverter *ex)

  Converter's external subplugins should call this at init.
- void unregisterExternalConverter (const char *name)

  Converter's external subplugins should call this at exit.
- void nnstreamer_converter_set_custom_property_desc (const char *name, const char *prop,...)

  set custom property description for tensor converter sub-plugin
- int nnstreamer_converter_custom_register (const gchar *name, tensor_converter_custom func, void *data)

  Registers a callback for tensor_converter custom condition.
- int nnstreamer_converter_custom_unregister (const gchar *name)

  Unregisters a callback for tensor_converter custom condition.

9.26.1 Detailed Description

GStreamer plugin to convert media types to tensors (as a filter for other general neural network filters)

GStreamer Copyright (C) 2005 Thomas Vander Stichele thomas@apestaart.org Copyright (C) 2005 Ronald S. Bultje rbultje@ronald.bitfreak.net Copyright (C) 2018 MyungJoo Ham myungjoo.ham@samsung.com

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Library General Public License as published by the Free Software Foundation; version 2.1 of the License.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Library General Public License for more details.

Date

26 Mar 2018

See also

https://github.com/nnstreamer/nnstreamer

Author

MyungJoo Ham myungjoo.ham@samsung.com

Bug No known bugs except for NYI items

Todo For flatbuffers, support other/tensors with properties

Subplugins are not tested, yet.

9.26.2 Macro Definition Documentation

9.26.2.1 append_flex_tensor_caps_template

```
#define append_flex_tensor_caps_template(
              caps ) gst_caps_append (caps, gst_caps_from_string (GST_TENSORS_FLEX_CAP_DEFAULT))
```

Macro to append template caps for flexible tensor.

Definition at line 85 of file gsttensor_converter.c.

9.26.2.2 append_octet_caps_template

```
#define append_octet_caps_template(
              caps ) gst_caps_append (caps, gst_caps_from_string (OCTET_CAPS_STR))
```

Definition at line 79 of file gsttensor_converter.c.

9.26.2.3 append_text_caps_template

```
#define append_text_caps_template(
              caps ) gst_caps_append (caps, gst_caps_from_string (TEXT_CAPS_STR))
```

Definition at line 71 of file gsttensor_converter.c.

9.26.2.4 DBG

```
#define DBG (!self->silent)
```

Macro for debug mode.

Definition at line 92 of file gsttensor_converter.c.

9.26.2.5 DEFAULT_FRAMES_PER_TENSOR

```
#define DEFAULT_FRAMES_PER_TENSOR 1
```

Frames in output tensor.

Definition at line 142 of file gsttensor_converter.c.

9.26.2.6 DEFAULT_SET_TIMESTAMP

```
#define DEFAULT_SET_TIMESTAMP TRUE
```

Flag to set timestamp when received a buffer with invalid timestamp.

Definition at line 132 of file gsttensor_converter.c.

9.26.2.7 DEFAULT_SILENT

```
#define DEFAULT_SILENT TRUE
```

Flag to print minimized log.

Definition at line 137 of file gsttensor_converter.c.

9.26.2.8 GST_CAT_DEFAULT

```
#define GST_CAT_DEFAULT gst_tensor_converter_debug
```

Definition at line 104 of file gsttensor_converter.c.

9.26.2.9 gst_tensor_converter_parent_class

```
#define gst_tensor_converter_parent_class parent_class
```

Definition at line 144 of file gsttensor_converter.c.

9.26.2.10 OCTET_CAPS_STR

```
#define OCTET_CAPS_STR "application/octet-stream"
```

Caps string for binary stream.

Definition at line 77 of file gsttensor_converter.c.

9.26.2.11 silent_debug_timestamp

```
#define silent_debug_timestamp(
              self,
              buf )
```

Value:

```
  do { \
  if (DBG) { \
    GST_DEBUG_OBJECT (self, "pts = %" GST_TIME_FORMAT, GST_TIME_ARGS (GST_BUFFER_PTS (buf))); \
    GST_DEBUG_OBJECT (self, "dts = %" GST_TIME_FORMAT, GST_TIME_ARGS (GST_BUFFER_DTS (buf))); \
    GST_DEBUG_OBJECT (self, "duration = %" GST_TIME_FORMAT "\n", GST_TIME_ARGS (GST_BUFFER_DURATION (buf)));
      \
  } \
} while (0)
```

Definition at line 95 of file gsttensor_converter.c.

9.26.2.12 STRING_CUSTOM_MODE

```
#define STRING_CUSTOM_MODE(
              self )
```

Value:

```
    (((self)->mode == _CONVERTER_MODE_CUSTOM_CODE) ? \
        "custom_code (function)" : \
        (((self)->mode == _CONVERTER_MODE_CUSTOM_SCRIPT) ? \
            "custom_script (py)" : \
            "unknown custom mode (internal error!)"))
```

Definition at line 106 of file gsttensor_converter.c.

9.26.2.13 TEXT_CAPS_STR

```
#define TEXT_CAPS_STR "text/x-raw, format = (string) utf8"
```

Caps string for text input.

SECTION:element-tensor_converter

A filter that converts media stream to tensor stream for NN frameworks. The output is always in the format of other/tensor or other/tensors.

<refsect2> <title>Example launch line</title> |[gst-launch-1.0 videotestsrc ! video/x-raw,format=R↩ GB,width=640,height=480 ! tensor_converter ! tensor_sink]| </refsect2>

Definition at line 69 of file gsttensor_converter.c.

9.26.3 Enumeration Type Documentation

9.26.3.1 anonymous enum

```
anonymous enum
```

tensor_converter properties

Todo For flatbuffers, support other/tensors.

Enumerator

PROP_0	
PROP_INPUT_DIMENSION	
PROP_INPUT_TYPE	
PROP_FRAMES_PER_TENSOR	
PROP_SET_TIMESTAMP	
PROP_SUBPLUGINS	
PROP_SILENT	
PROP_MODE	

Definition at line 117 of file gsttensor_converter.c.

9.26.4 Function Documentation

9.26.4.1 _gst_tensor_converter_chain_chunk()

```
static GstFlowReturn _gst_tensor_converter_chain_chunk (
            GstTensorConverter * self,
            GstBuffer * inbuf,
            guint frames_in,
            guint frames_out,
            gsize frame_size )  [static]
```

Chain function's private routine to push multiple buffers.

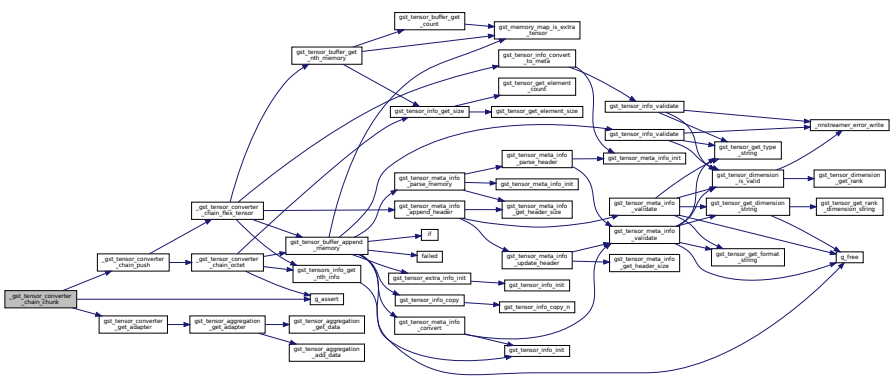
supposed same duration for incoming buffer

Update timestamp. If frames-in is larger then frames-out, the same timestamp (pts and dts) would be returned.

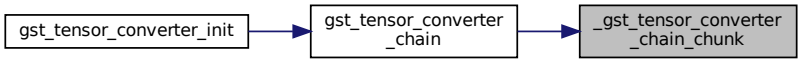
set timestamp

Definition at line 959 of file gsttensor_converter.c.

Here is the call graph for this function:



Here is the caller graph for this function:



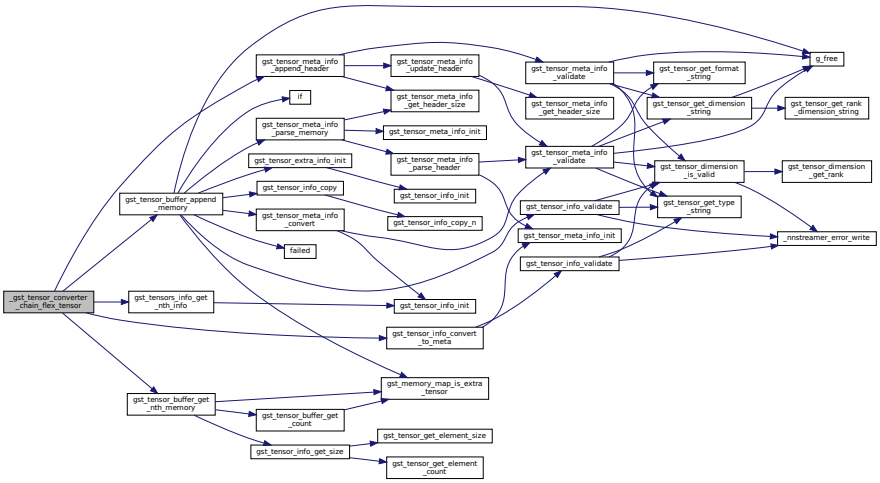
9.26.4.2 _gst_tensor_converter_chain_flex_tensor()

```
static GstBuffer* _gst_tensor_converter_chain_flex_tensor (
            GstTensorConverter * self,
            GstBuffer * buf )  [static]
```

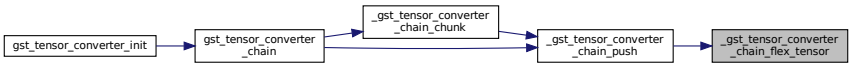
Chain function's private routine to process flex tensor.

Definition at line 893 of file gsttensor_converter.c.

Here is the call graph for this function:



Here is the caller graph for this function:



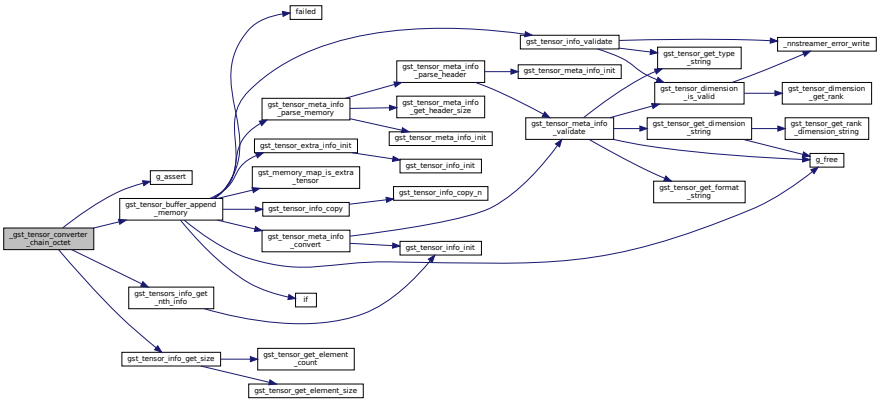
9.26.4.3 _gst_tensor_converter_chain_octet()

```
static GstBuffer* _gst_tensor_converter_chain_octet (
            GstTensorConverter * self,
            GstBuffer * buf )  [static]
```

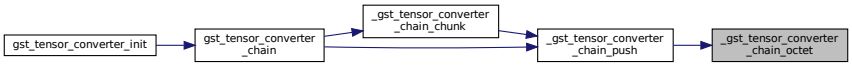
Chain function's private routine to process octet stream.

Definition at line 847 of file gsttensor_converter.c.

Here is the call graph for this function:



Here is the caller graph for this function:



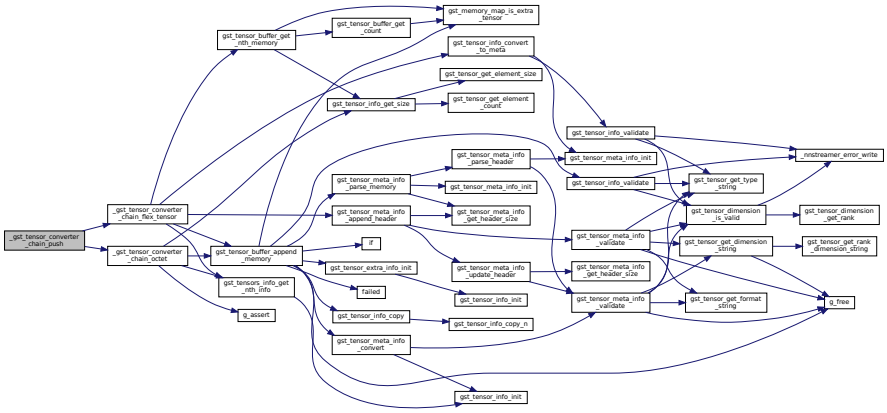
9.26.4.4 _gst_tensor_converter_chain_push()

```
static GstFlowReturn _gst_tensor_converter_chain_push (
            GstTensorConverter * self,
            GstBuffer * buf )  [static]
```

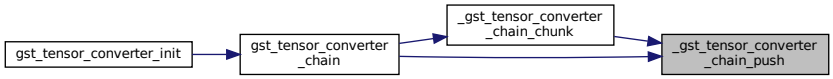
Chain function's private routine to push buffer into src pad.

Definition at line 938 of file gsttensor_converter.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.26.4.5 _gst_tensor_converter_chain_segment()

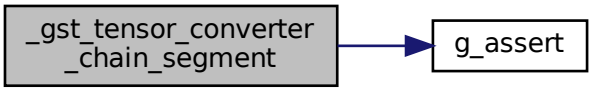
```
static void _gst_tensor_converter_chain_segment (
            GstTensorConverter * self,
            gsize frame_size )  [static]
```

Chain function's private routine.

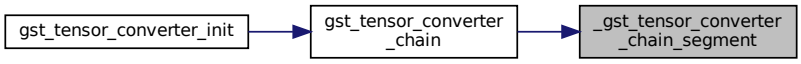
This is an internal logic error.

Definition at line 754 of file gsttensor_converter.c.

Here is the call graph for this function:



Here is the caller graph for this function:



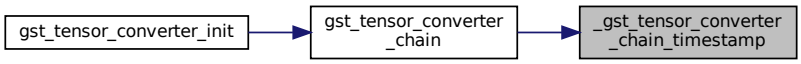
9.26.4.6 _gst_tensor_converter_chain_timestamp()

```
static void _gst_tensor_converter_chain_timestamp (
            GstTensorConverter * self,
            GstBuffer * inbuf,
            guint frames_in )  [static]
```

Chain function's private routine.

Definition at line 787 of file gsttensor_converter.c.

Here is the caller graph for this function:



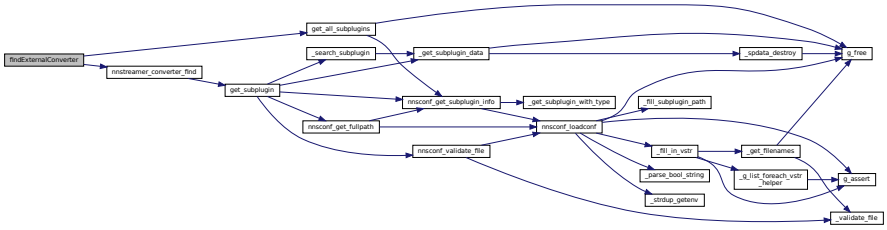
9.26.4.7 findExternalConverter()

```
static const NNStreamerExternalConverter * findExternalConverter (
            const char * media_type_name )  [static]
```

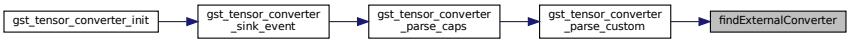
Internal static function to find registered subplugins.

Definition at line 2408 of file gsttensor_converter.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.26.4.8 G_DEFINE_TYPE()

```
G_DEFINE_TYPE (
              GstTensorConverter ,
              gst_tensor_converter ,
              GST_TYPE_ELEMENT  )
```

9.26.4.9 GST_DEBUG_CATEGORY_STATIC()

```
GST_DEBUG_CATEGORY_STATIC (
              gst_tensor_converter_debug  )
```

9.26.4.10 gst_tensor_converter_chain()

```
static GstFlowReturn gst_tensor_converter_chain (
              GstPad * pad,
              GstObject * parent,
              GstBuffer * buf )  [static]
```

Chain function, this function does the actual processing.

This is an internal logic error.

Supposed 1 frame in buffer (default). Update frame size for each media type.

type ∗ colorspace ∗ width ∗ height

supposed 1 frame in buffer

Refer: https://gstreamer.freedesktop.org/documentation/design/mediatype-video-raw.html

Internal logic error!

copy timestamps

copy timestamps

Todo need rewrite. do not use assert

Todo expand mem if given property is larger than mem size. Now compare same size, later we should modify mem block if developer sets different dimension.

Todo identify and printout the given input stream caps.

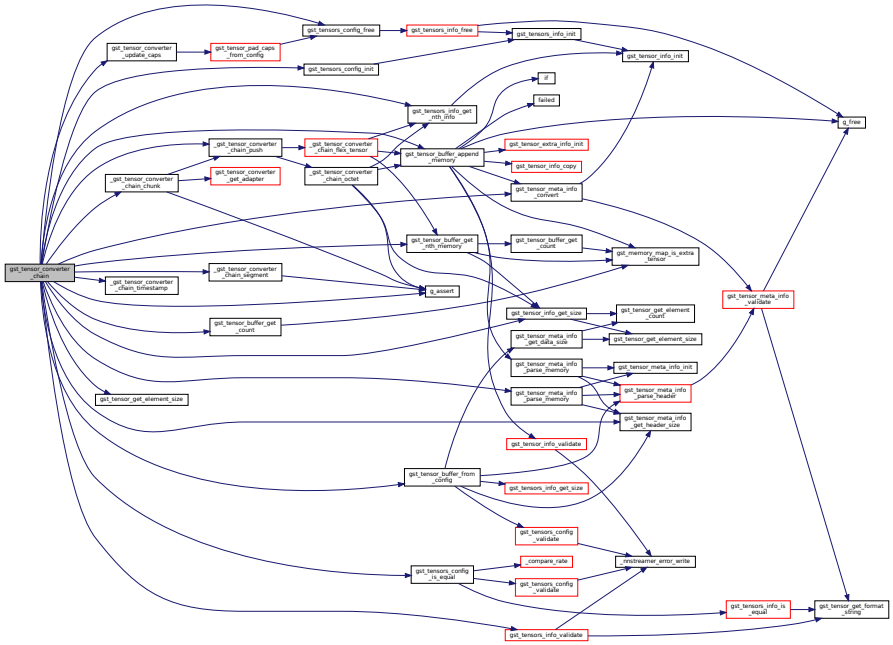
convert format (bytes > time) and push segment event. It will push event if needed (self->need_segment is true).

configures timestamp if required (self->set_timestamp is true)

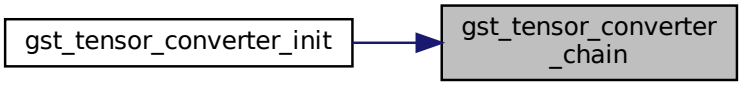
do nothing, push the incoming buffer

Definition at line 1028 of file gsttensor_converter.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.26.4.11 gst_tensor_converter_change_state()

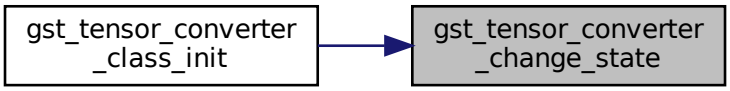
```
static GstStateChangeReturn gst_tensor_converter_change_state (
            GstElement * element,
            GstStateChange transition )  [static]
```

Called to perform state change.

Definition at line 1336 of file gsttensor_converter.c.

Here is the call graph for this function:

Here is the caller graph for this function:



9.26.4.12 gst_tensor_converter_class_init()

```
static void gst_tensor_converter_class_init (
            GstTensorConverterClass * klass )  [static]
```

Initialize the tensor_converter's class.

GstTensorConverter::input-dim:

Input tensor dimension from inner array. Generally this property is used to set tensor configuration for byte-stream (application/octet-stream). When setting this property and input media type is video or audio stream, GstTensor↩ Converter will compare the media info with this. (If it is different, it will be failed.)

GstTensorConverter::input-type:

Type of each element of the input tensor. Generally this property is used to set tensor configuration for byte-stream (application/octet-stream). When setting this property and input media type is video or audio stream, GstTensor↩ Converter will compare the media info with this. (If it is different, it will be failed.)

GstTensorConverter::frames-per-tensor:

The number of frames in outgoing buffer. (buffer is a single tensor instance) GstTensorConverter can push a buffer with multiple media frames.

GstTensorConverter::set-timestamp:

The flag to set timestamp when received a buffer with invalid timestamp.

GstTensorConverter::sub-plugins:

Registrable sub-plugins list of tensor-converter.

GstTensorConverter::silent:

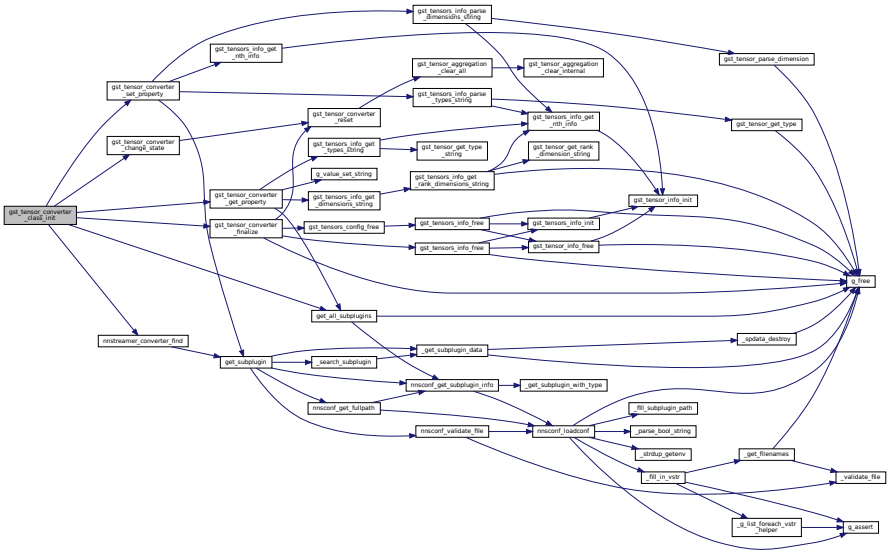
The flag to enable/disable debugging messages.

GstTensorConverter::mode:

Generally this property is used to set tensor converter custom mode.

Definition at line 178 of file gsttensor_converter.c.

Here is the call graph for this function:



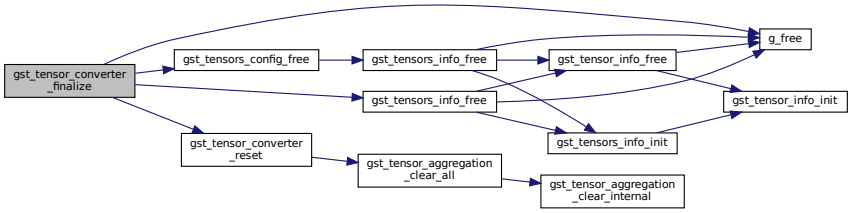
9.26.4.13 gst_tensor_converter_finalize()

```
static void gst_tensor_converter_finalize (
            GObject * object )  [static]
```

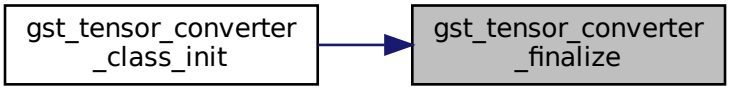
Function to finalize instance.

Definition at line 381 of file gsttensor_converter.c.

Here is the call graph for this function:



Here is the caller graph for this function:



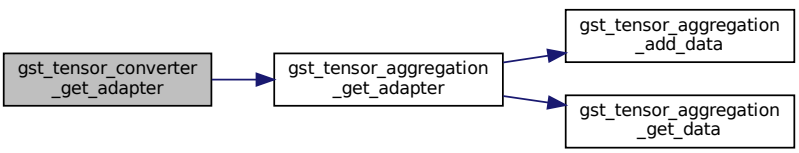
9.26.4.14 gst_tensor_converter_get_adapter()

```
static GstAdapter* gst_tensor_converter_get_adapter (
            GstTensorConverter * self,
            GstBuffer * buf )  [static]
```

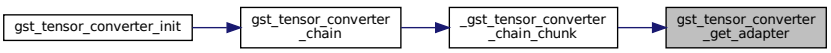
Internal function to get adapter.

Definition at line 706 of file gsttensor_converter.c.

Here is the call graph for this function:



Here is the caller graph for this function:



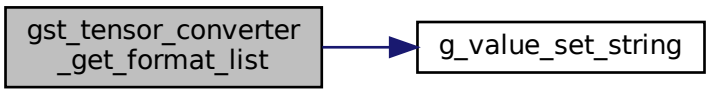
9.26.4.15 gst_tensor_converter_get_format_list()

```
static void gst_tensor_converter_get_format_list (
             GValue * list,
              ...  )  [static]
```

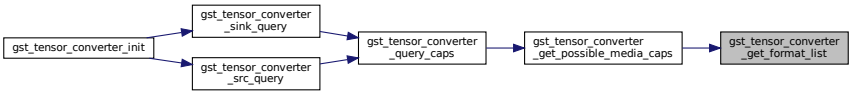
Get supported format list.

Definition at line 1385 of file gsttensor_converter.c.

Here is the call graph for this function:



Here is the caller graph for this function:



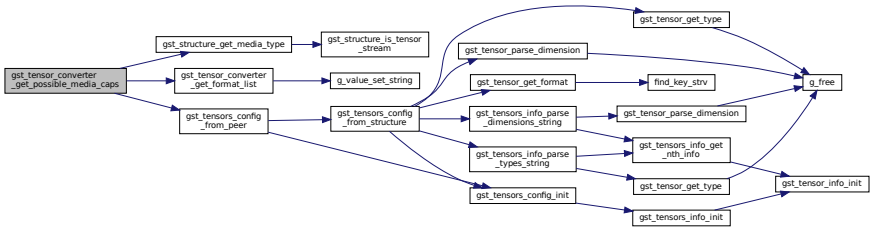
9.26.4.16 gst_tensor_converter_get_possible_media_caps()

```
static GstCaps* gst_tensor_converter_get_possible_media_caps (
            GstTensorConverter * self )  [static]
```

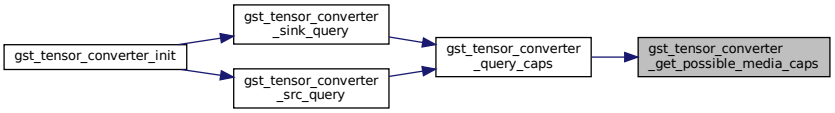
Get possible media-caps from downstream element.

Definition at line 1972 of file gsttensor_converter.c.

Here is the call graph for this function:



Here is the caller graph for this function:



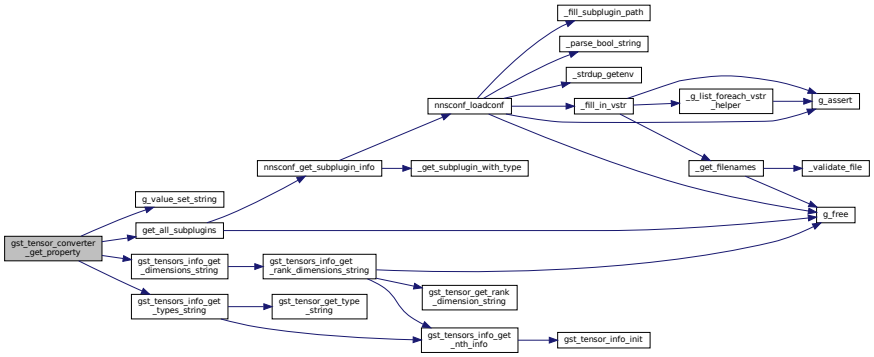
9.26.4.17 gst_tensor_converter_get_property()

```
static void gst_tensor_converter_get_property (
            GObject * object,
            guint prop_id,
            GValue * value,
            GParamSpec * pspec )  [static]
```

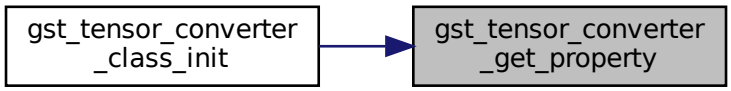
Getter for tensor_converter properties.

Definition at line 514 of file gsttensor_converter.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.26.4.18 gst_tensor_converter_init()

```
static void gst_tensor_converter_init (
            GstTensorConverter * self )  [static]
```

Initialize tensor_converter element.

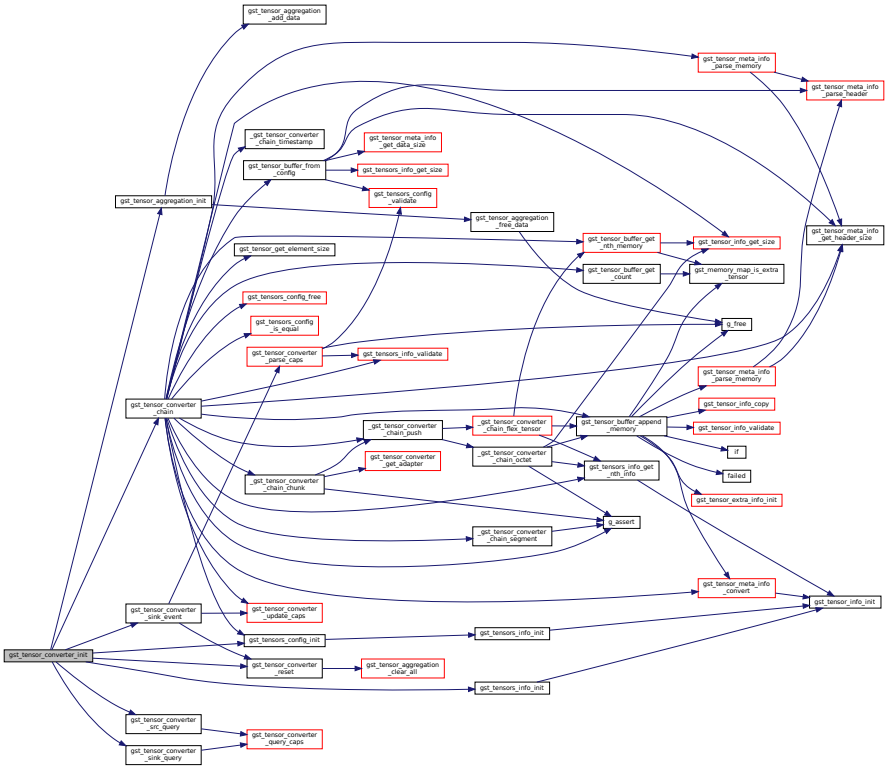
setup sink pad

setup src pad

init properties

Definition at line 331 of file gsttensor_converter.c.

Here is the call graph for this function:



9.26.4.19 gst_tensor_converter_parse_audio()

```
static gboolean gst_tensor_converter_parse_audio (
            GstTensorConverter * self,
            const GstCaps * caps,
            GstTensorsConfig * config )  [static]
```

Set the tensors config structure from audio info (internal static function)

Parameters

self	this pointer to GstTensorConverter
caps	caps for media stream
config	tensors config structure to be filled

Note

Change dimension if tensor contains N frames.

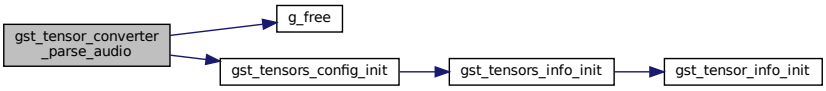
Returns

TRUE if supported type

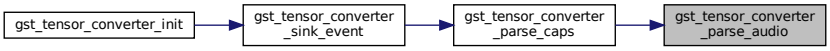
Refer: https://www.tensorflow.org/api_docs/python/tf/summary/audio A 3-D float32 Tensor of shape [batch_size, frames, channels] or a 2-D float32 Tensor of shape [batch_size, frames].

Definition at line 1582 of file gsttensor_converter.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.26.4.20 gst_tensor_converter_parse_caps()

```
static gboolean gst_tensor_converter_parse_caps (
            GstTensorConverter * self,
            const GstCaps * caps )  [static]
```

Parse caps and set tensors info.

dimension index of frames in configured tensors

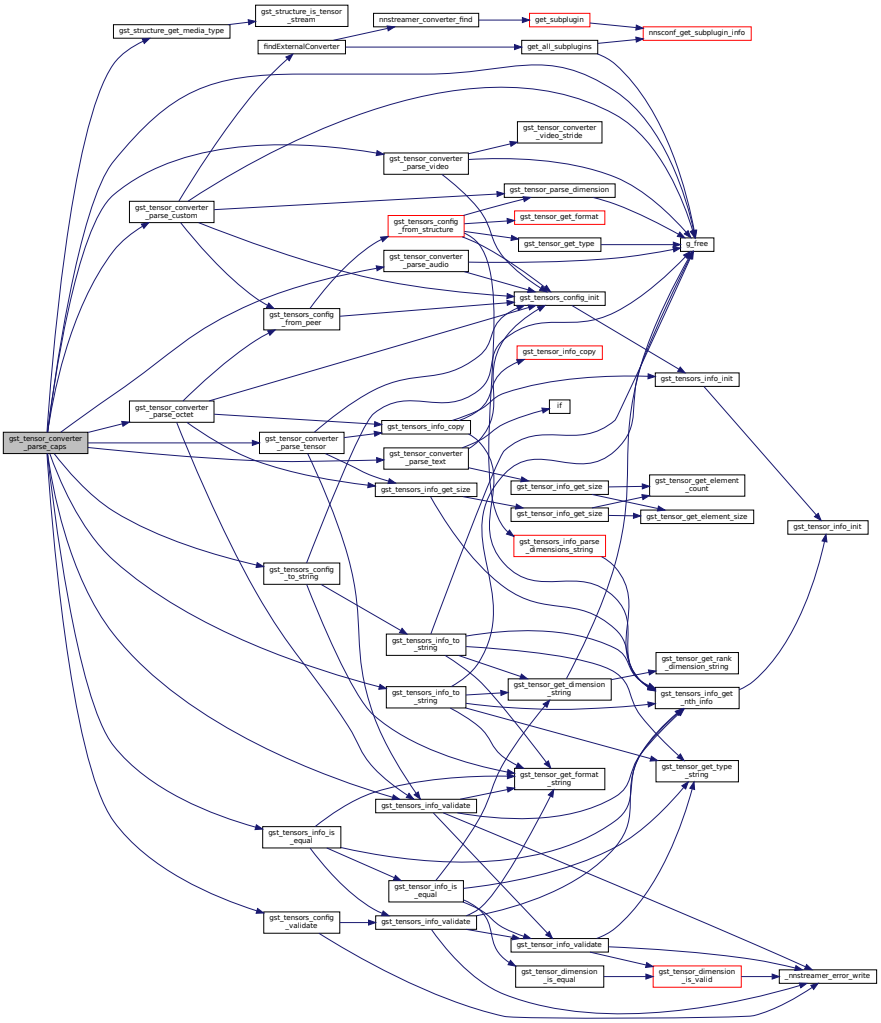
set the number of frames in dimension

not fully configured. the resulting config is weird.

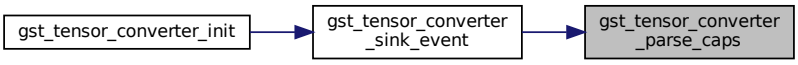
compare tensor info

Definition at line 2190 of file gsttensor_converter.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.26.4.21 gst_tensor_converter_parse_custom()

```
static gboolean gst_tensor_converter_parse_custom (
            GstTensorConverter * self,
            GstTensorsConfig * config,
            const GstCaps * caps )  [static]
```

Set the tensors config structure from caps (internal static function for custom mode)

Parameters

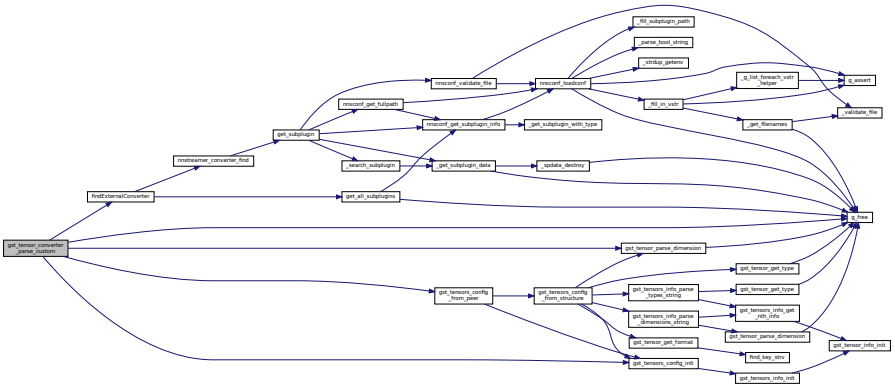
self	this pointer to GstTensorConverter
config	tensors config structure to be filled
caps	incoming caps

Returns

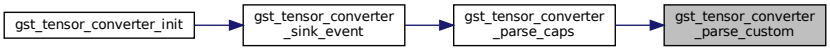
TRUE if supported type

Definition at line 1889 of file gsttensor_converter.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.26.4.22 gst_tensor_converter_parse_octet()

```
static gboolean gst_tensor_converter_parse_octet (
            GstTensorConverter * self,
            GstTensorsConfig * config,
            const GstStructure * structure )  [static]
```

Set the tensors configs structure from octet stream (internal static function)

Parameters

self	this pointer to GstTensorConverter
config	tensors config structure to be filled
structure	caps structure

Note

Change tensors dimension and type.

Returns

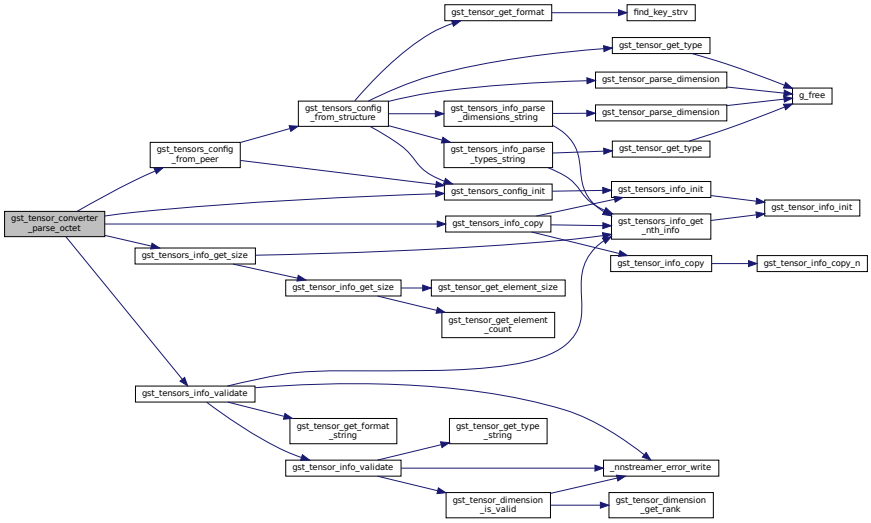
TRUE if supported type

Failure case when octet-stream has multi tensors and multi frames.

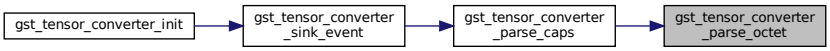
Raw byte-stream (application/octet-stream) We cannot get the exact tensors info from caps. All tensors info should be updated. If output is flexible, dimension should be updated in chain function with buffer size. (data format for tensor: [size])

Definition at line 1741 of file gsttensor_converter.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.26.4.23 gst_tensor_converter_parse_tensor()

```
static gboolean gst_tensor_converter_parse_tensor (
            GstTensorConverter * self,
            GstTensorsConfig * config,
            const GstStructure * structure )  [static]
```

Set the tensors configs structure from fliex tensor stream (internal static function)

Parameters

self	this pointer to GstTensorConverter
config	tensors config structure to be filled
structure	caps structure

Note

Change tensors dimension and type.

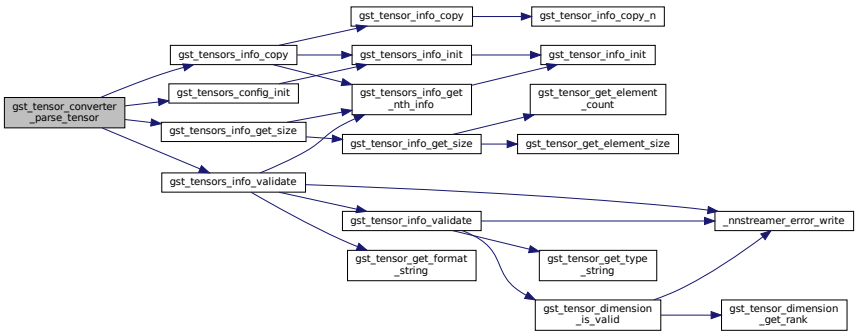
Returns

TRUE if supported type

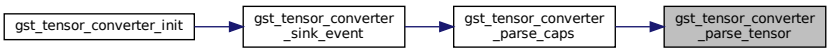
We cannot get the exact tensors info from caps. All tensors info should be updated in chain function. (data format for tensor: [size])

Definition at line 1834 of file gsttensor_converter.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.26.4.24 gst_tensor_converter_parse_text()

```
static gboolean gst_tensor_converter_parse_text (
            GstTensorConverter * self,
            GstTensorsConfig * config,
            const GstStructure * structure )  [static]
```

Set the tensors config structure from text info (internal static function)

Parameters

self	this pointer to GstTensorConverter
config	tensors config structure to be filled
structure	caps structure

Note

Change dimension if tensors contains N frames.

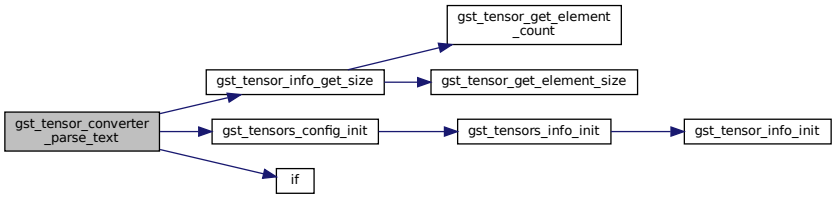
Returns

TRUE if supported type

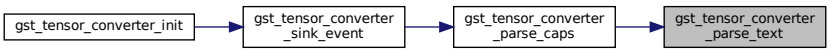
Refer: https://www.tensorflow.org/api_docs/python/tf/summary/text A string-type Tensor

Definition at line 1670 of file gsttensor_converter.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.26.4.25 gst_tensor_converter_parse_video()

```
static gboolean gst_tensor_converter_parse_video (
            GstTensorConverter * self,
            const GstCaps * caps,
            GstTensorsConfig * config )  [static]
```

Set the tensors config structure from video info (internal static function)

Parameters

self	this pointer to GstTensorConverter
caps	caps for media stream
config	tensors config structure to be filled

Note

Change dimension if tensor contains N frames.

Returns

TRUE if supported type

Refer: https://www.tensorflow.org/api_docs/python/tf/summary/image A 4-D uint8 or float32 Tensor of shape [batch_size, height, width, channels] where channels is 1, 3, or 4.

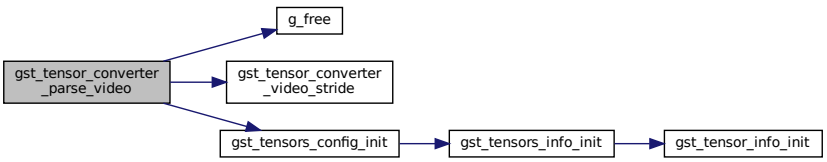
Emit Warning if RSTRIDE = RU4 (3BPP) && Width % 4 > 0

Todo Add more conditions!

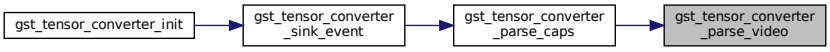
Todo need rewrite.

Definition at line 1444 of file gsttensor_converter.c.

Here is the call graph for this function:



Here is the caller graph for this function:



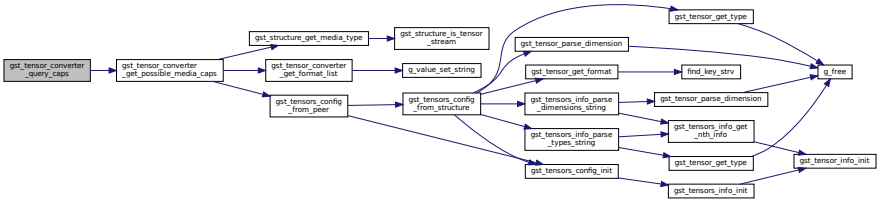
9.26.4.26 gst_tensor_converter_query_caps()

```
static GstCaps * gst_tensor_converter_query_caps (
            GstTensorConverter * self,
            GstPad * pad,
            GstCaps * filter )  [static]
```

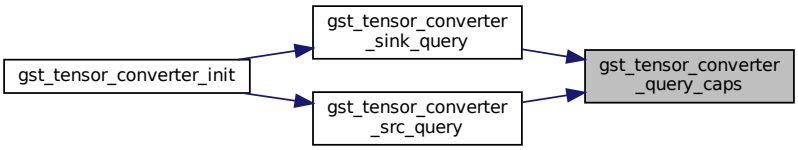
Get pad caps for caps negotiation.

Definition at line 2144 of file gsttensor_converter.c.

Here is the call graph for this function:



Here is the caller graph for this function:



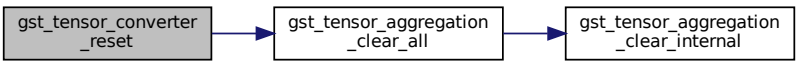
9.26.4.27 gst_tensor_converter_reset()

```
static void gst_tensor_converter_reset (
            GstTensorConverter * self )  [static]
```

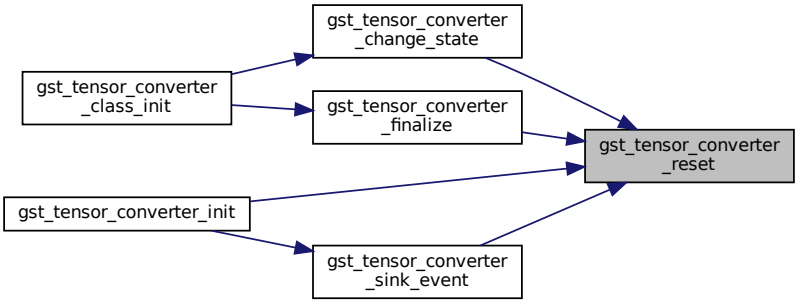
Clear and reset data.

Definition at line 1369 of file gsttensor_converter.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.26.4.28 gst_tensor_converter_set_property()

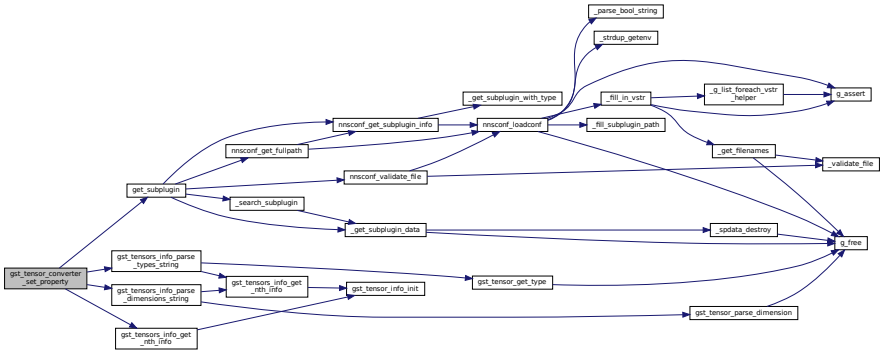
```
static void gst_tensor_converter_set_property (
            GObject * object,
            guint prop_id,
            const GValue * value,
            GParamSpec * pspec )  [static]
```

Setter for tensor_converter properties.

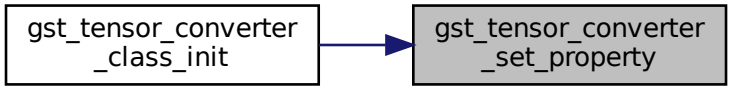
Todo detects framework based on the script extension

Definition at line 406 of file gsttensor_converter.c.

Here is the call graph for this function:



Here is the caller graph for this function:



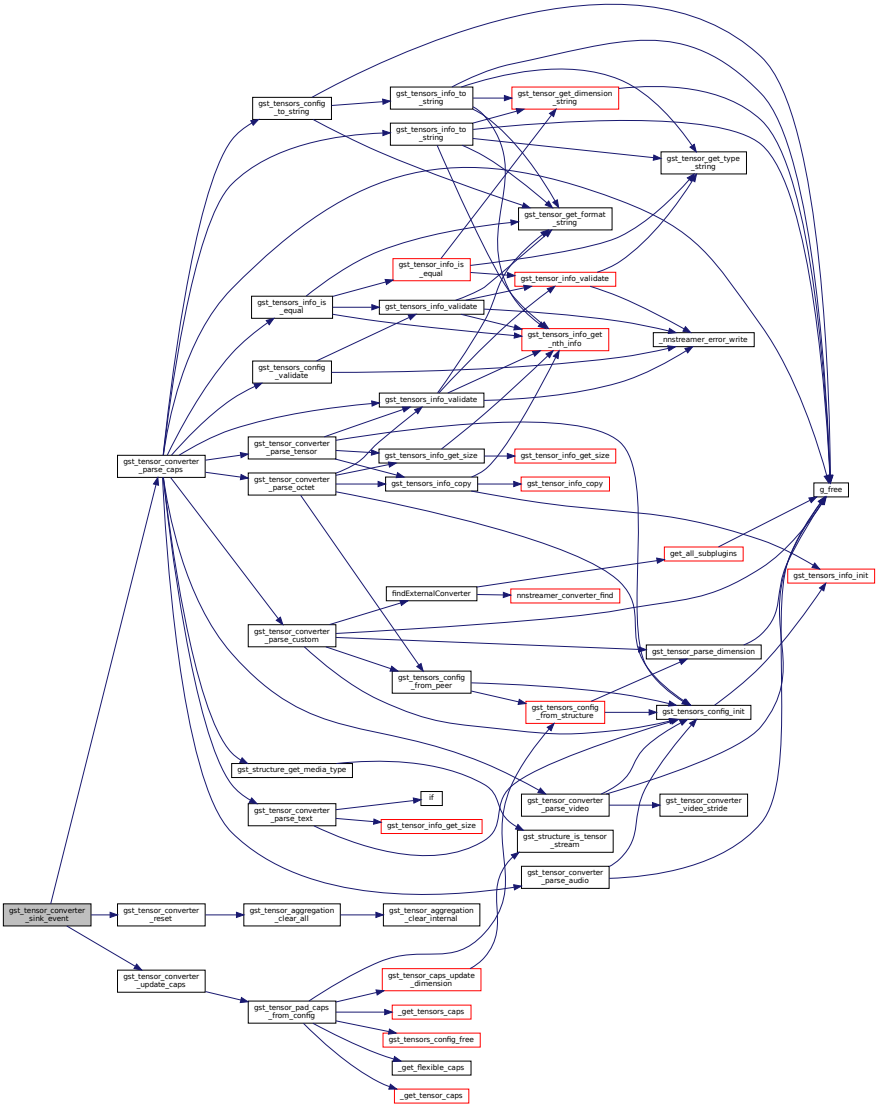
9.26.4.29 gst_tensor_converter_sink_event()

```
static gboolean gst_tensor_converter_sink_event (
             GstPad * pad,
             GstObject * parent,
             GstEvent * event )  [static]
```

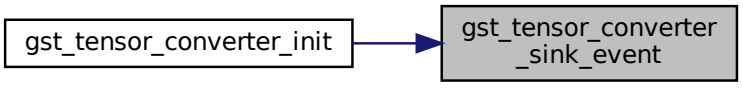
This function handles sink event.

Definition at line 586 of file gsttensor_converter.c.

Here is the call graph for this function:



Here is the caller graph for this function:



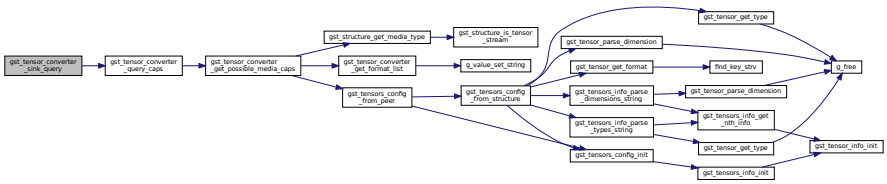
9.26.4.30 gst_tensor_converter_sink_query()

```
static gboolean gst_tensor_converter_sink_query (
            GstPad * pad,
            GstObject * parent,
            GstQuery * query )  [static]
```

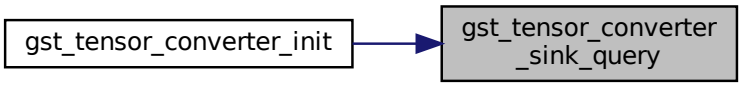
This function handles sink pad query.

Definition at line 654 of file gsttensor_converter.c.

Here is the call graph for this function:



Here is the caller graph for this function:



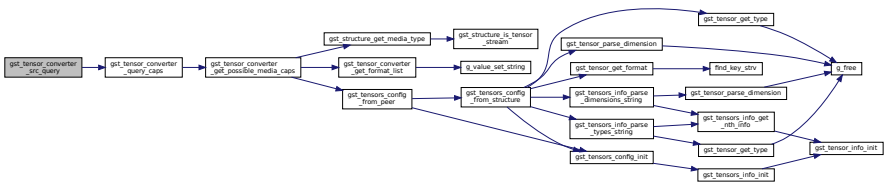
9.26.4.31 gst_tensor_converter_src_query()

```
static gboolean gst_tensor_converter_src_query (
            GstPad * pad,
            GstObject * parent,
            GstQuery * query )  [static]
```

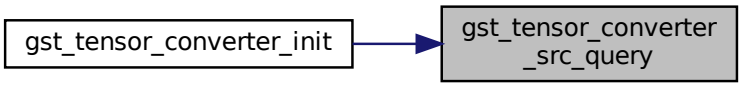
This function handles src pad query.

Definition at line 722 of file gsttensor_converter.c.

Here is the call graph for this function:



Here is the caller graph for this function:



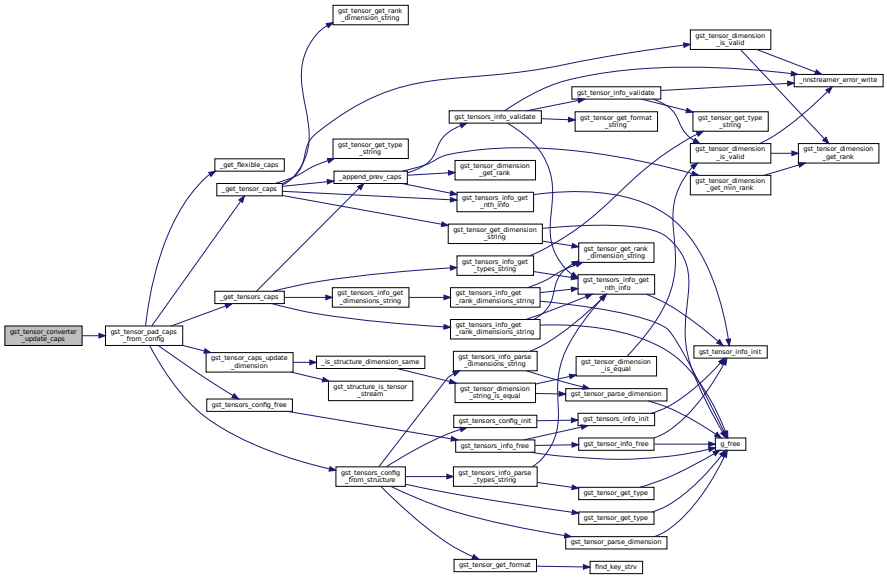
9.26.4.32 gst_tensor_converter_update_caps()

```
static void gst_tensor_converter_update_caps (
            GstTensorConverter * self )  [static]
```

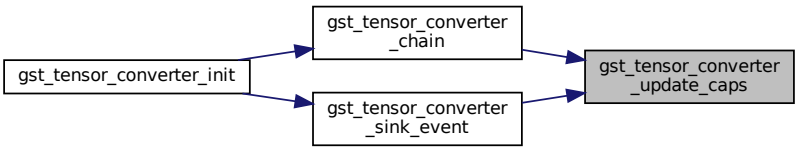
Update src pad caps from tensors config.

Definition at line 2333 of file gsttensor_converter.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.26.4.33 gst_tensor_converter_video_stride()

```
static gboolean gst_tensor_converter_video_stride (
            GstVideoFormat format,
            gint width )  [static]
```

Determine if we need zero-padding.

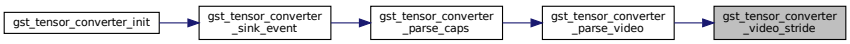
Returns

TRUE if we need to add (or remove) stride per row from the stream data.

Todo The actual list is much longer, fill them. (read https://gstreamer.freedesktop.org/documentation/design/mediatype-video-raw.html)

Definition at line 1409 of file gsttensor_converter.c.

Here is the caller graph for this function:



9.26.4.34 nnstreamer_converter_custom_register()

```
int nnstreamer_converter_custom_register (
            const gchar * name,
            tensor_converter_custom func,
            void * data )
```

Registers a callback for tensor_converter custom condition.

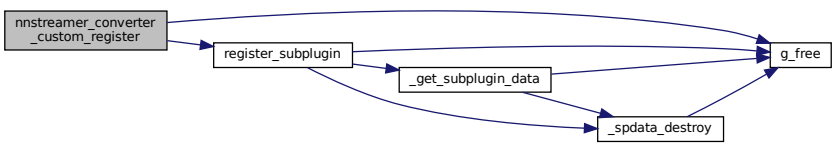
Register the custom callback function.

Returns

0 if success. -ERRNO if error.

Definition at line 2473 of file gsttensor_converter.c.

Here is the call graph for this function:



9.26.4.35 nnstreamer_converter_custom_unregister()

```
int nnstreamer_converter_custom_unregister (
            const gchar * name )
```

Unregisters a callback for tensor_converter custom condition.

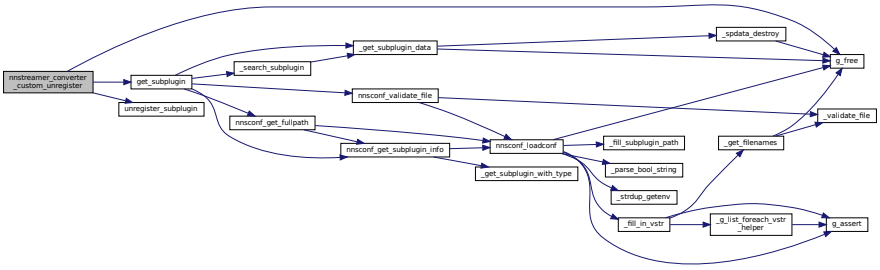
Unregister the custom callback function.

Returns

0 if success. -ERRNO if error.

Definition at line 2500 of file gsttensor_converter.c.

Here is the call graph for this function:



9.26.4.36 nnstreamer_converter_find()

```
const NNStreamerExternalConverter* nnstreamer_converter_find (
            const char * name )
```

Find converter sub-plugin with the name.

Parameters

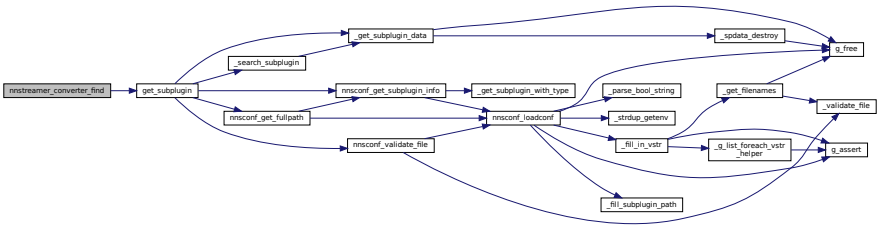
in	*name*	The name of converter sub-plugin.

Returns

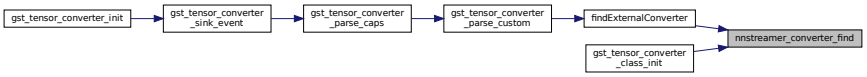
NULL if not found or the sub-plugin object has an error.

Definition at line 2360 of file gsttensor_converter.c.

Here is the call graph for this function:



Here is the caller graph for this function:



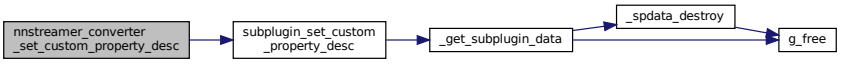
9.26.4.37 nnstreamer_converter_set_custom_property_desc()

```
void nnstreamer_converter_set_custom_property_desc (
            const char * name,
            const char * prop,
             ...  )
```

set custom property description for tensor converter sub-plugin

Definition at line 2457 of file gsttensor_converter.c.

Here is the call graph for this function:



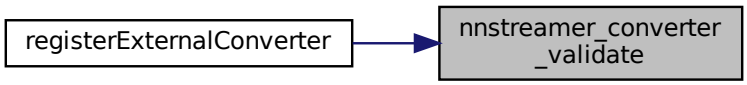
9.26.4.38 nnstreamer_converter_validate()

```
static gboolean nnstreamer_converter_validate (
            const NNStreamerExternalConverter * converter )  [static]
```

Validate converter sub-plugin's data.

Definition at line 2369 of file gsttensor_converter.c.

Here is the caller graph for this function:



9.26.4.39 registerExternalConverter()

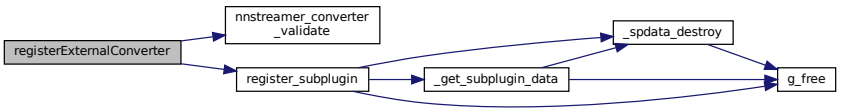
```
int registerExternalConverter (
            NNStreamerExternalConverter * ex )
```

Converter's external subplugins should call this at init.

Converter's sub-plugin should call this function to register itself.

Definition at line 2389 of file gsttensor_converter.c.

Here is the call graph for this function:



9.26.4.40 unregisterExternalConverter()

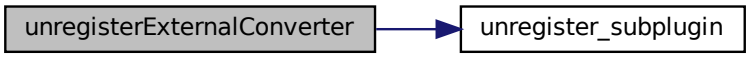
```
void unregisterExternalConverter (
            const char * name )
```

Converter's external subplugins should call this at exit.

Converter's sub-plugin may call this to unregister itself.

Definition at line 2399 of file gsttensor_converter.c.

Here is the call graph for this function:

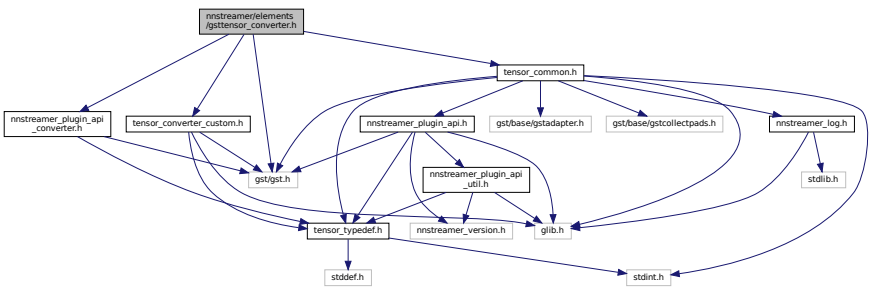


9.27 nnstreamer/elements/gsttensor_converter.h File Reference

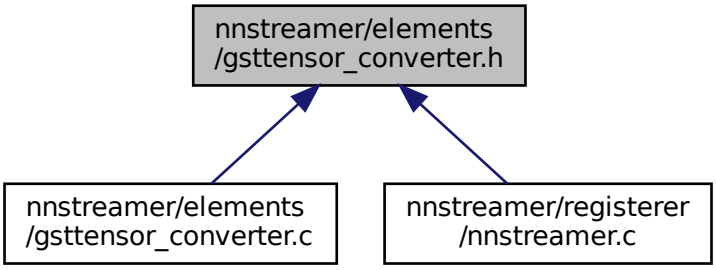
GStreamer plugin to convert media types to tensors (as a filter for other general neural network filters)

```
#include <gst/gst.h>
#include <tensor_common.h>
#include "nnstreamer_plugin_api_converter.h"
#include "tensor_converter_custom.h"
```

Include dependency graph for gsttensor_converter.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct converter_custom_cb_s
- struct _GstTensorConverter

  Internal data structure for tensor_converter instances.
- struct _GstTensorConverterClass

  GstTensorConverterClass data structure.

Macros

- #define GST_TYPE_TENSOR_CONVERTER (gst_tensor_converter_get_type())
- #define GST_TENSOR_CONVERTER(obj) (G_TYPE_CHECK_INSTANCE_CAST((obj),GST_TYPE_TENSOR_CONVERTER
- #define GST_TENSOR_CONVERTER_CLASS(klass) (G_TYPE_CHECK_CLASS_CAST((klass),GST_TYPE_TENSOR_CON
- #define GST_IS_TENSOR_CONVERTER(obj) (G_TYPE_CHECK_INSTANCE_TYPE((obj),GST_TYPE_TENSOR_CONVERT
- #define GST_IS_TENSOR_CONVERTER_CLASS(klass) (G_TYPE_CHECK_CLASS_TYPE((klass),GST_TYPE_TENSOR_C

Typedefs

- typedef struct _GstTensorConverter GstTensorConverter
- typedef struct _GstTensorConverterClass GstTensorConverterClass

Enumerations

- enum tensor_converter_mode { _CONVERTER_MODE_NONE = 0, _CONVERTER_MODE_CUSTOM_CODE = 1, _CONVERTER_MODE_CUSTOM_SCRIPT = 2 }

  tensor converter mode

Functions

- GType gst_tensor_converter_get_type (void)

  Get Type function required for gst elements.

9.27.1 Detailed Description

GStreamer plugin to convert media types to tensors (as a filter for other general neural network filters)

Date

26 Mar 2018

```
 Be careful: this filter assumes that the user has attached
other GST converters as a preprocessor for this filter so that
the incoming buffer is nicely aligned in the array of
uint8[height][width][RGB]. Note that if rstride=RU4, you need
to add the case in "remove_stride_padding_per_row".
```

See also

https://github.com/nnstreamer/nnstreamer

Author

MyungJoo Ham myungjoo.ham@samsung.com

Bug No known bugs except for NYI items

9.27.2 Macro Definition Documentation

9.27.2.1 GST_IS_TENSOR_CONVERTER

```
#define GST_IS_TENSOR_CONVERTER(
              obj ) (G_TYPE_CHECK_INSTANCE_TYPE((obj),GST_TYPE_TENSOR_CONVERTER))
```

Definition at line 50 of file gsttensor_converter.h.

9.27.2.2 GST_IS_TENSOR_CONVERTER_CLASS

```
#define GST_IS_TENSOR_CONVERTER_CLASS(
              klass ) (G_TYPE_CHECK_CLASS_TYPE((klass),GST_TYPE_TENSOR_CONVERTER))
```

Definition at line 52 of file gsttensor_converter.h.

9.27.2.3 GST_TENSOR_CONVERTER

```
#define GST_TENSOR_CONVERTER(
              obj ) (G_TYPE_CHECK_INSTANCE_CAST((obj),GST_TYPE_TENSOR_CONVERTER,GstTensorConverter))
```

Definition at line 46 of file gsttensor_converter.h.

9.27.2.4 GST_TENSOR_CONVERTER_CLASS

```
#define GST_TENSOR_CONVERTER_CLASS(
              klass ) (G_TYPE_CHECK_CLASS_CAST((klass),GST_TYPE_TENSOR_CONVERTER,GstTensorConverterClass))
```

Definition at line 48 of file gsttensor_converter.h.

9.27.2.5 GST_TYPE_TENSOR_CONVERTER

```
#define GST_TYPE_TENSOR_CONVERTER (gst_tensor_converter_get_type())
```

Definition at line 44 of file gsttensor_converter.h.

9.27.3 Typedef Documentation

9.27.3.1 GstTensorConverter

```
typedef struct _GstTensorConverter GstTensorConverter
```

Definition at line 55 of file gsttensor_converter.h.

9.27.3.2 GstTensorConverterClass

```
typedef struct _GstTensorConverterClass GstTensorConverterClass
```

Definition at line 56 of file gsttensor_converter.h.

9.27.4 Enumeration Type Documentation

9.27.4.1 tensor_converter_mode

```
enum tensor_converter_mode
```

tensor converter mode

Enumerator

_CONVERTER_MODE_NONE	Normal mode (default)
_CONVERTER_MODE_CUSTOM_CODE	Custom mode (callback type)
_CONVERTER_MODE_CUSTOM_SCRIPT	Custom mode (script type)

Definition at line 66 of file gsttensor_converter.h.

9.27.5 Function Documentation

9.27.5.1 gst_tensor_converter_get_type()

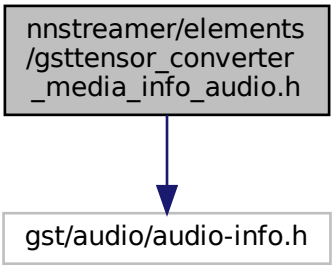
```
GType gst_tensor_converter_get_type (
            void  )
```

Get Type function required for gst elements.

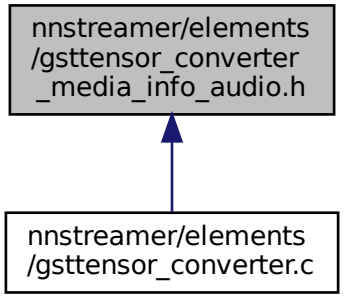
9.28 nnstreamer/elements/gsttensor_converter_media_info_audio.h File Reference

Define collection of media type and functions to parse media info for audio support.

```
#include <gst/audio/audio-info.h>
```
Include dependency graph for gsttensor_converter_media_info_audio.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define AUDIO_CAPS_STR

  Caps string for supported audio format.
- #define append_audio_caps_template(caps) gst_caps_append (caps, gst_caps_from_string (AUDIO_CAPS_STR))
- #define is_audio_supported(...) TRUE

9.28.1 Detailed Description

Define collection of media type and functions to parse media info for audio support.

NNStreamer media type definition for tensor-converter Copyright (C) 2019 Jijoong Moon jijoong.↩ moon@samsung.com

Date

26 Mar 2019

See also

https://github.com/nnstreamer/nnstreamer

Author

Jijoong Moon jijoong.moon@samsung.com

Bug No known bugs except for NYI items

9.28.2 Macro Definition Documentation

9.28.2.1 append_audio_caps_template

```
#define append_audio_caps_template(
              caps ) gst_caps_append (caps, gst_caps_from_string (AUDIO_CAPS_STR))
```

Definition at line 32 of file gsttensor_converter_media_info_audio.h.

9.28.2.2 AUDIO_CAPS_STR

```
#define AUDIO_CAPS_STR
```

Value:

```
GST_AUDIO_CAPS_MAKE ("{ S8, U8, S16LE, S16BE, U16LE, U16BE, S32LE, S32BE, U32LE, U32BE, F32LE, F32BE,
   F64LE, F64BE }") \
", layout = (string) interleaved"
```

Caps string for supported audio format.

Definition at line 28 of file gsttensor_converter_media_info_audio.h.

9.28.2.3 is_audio_supported

```
#define is_audio_supported(
              ... ) TRUE
```

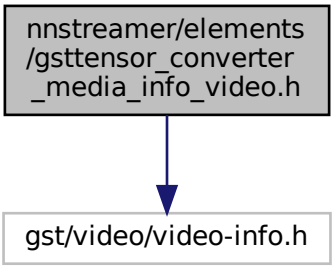
Definition at line 35 of file gsttensor_converter_media_info_audio.h.

9.29 nnstreamer/elements/gsttensor_converter_media_info_video.h File Reference

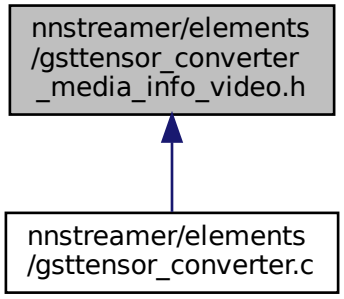
Define collection of media type and functions to parse media info for video support.

```
#include <gst/video/video-info.h>
```

Include dependency graph for gsttensor_converter_media_info_video.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define NNS_VIDEO_FORMAT "{ RGB, BGR, RGBx, BGRx, xRGB, xBGR, RGBA, BGRA, ARGB, ABGR, GRAY8, GRAY16_BE, GRAY16_LE }"

  Caps string for supported video format.
- #define VIDEO_CAPS_STR
- #define append_video_caps_template(caps) gst_caps_append (caps, gst_caps_from_string (VIDEO_CAPS_STR))
- #define is_video_supported(...) TRUE

9.29.1 Detailed Description

Define collection of media type and functions to parse media info for video support.

NNStreamer media type definition for tensor-converter Copyright (C) 2019 Jijoong Moon jijoong.↩ moon@samsung.com

Date

26 Mar 2019

See also

https://github.com/nnstreamer/nnstreamer

Author

Jijoong Moon jijoong.moon@samsung.com

Bug No known bugs except for NYI items

9.29.2 Macro Definition Documentation

9.29.2.1 append_video_caps_template

```
#define append_video_caps_template(
              caps ) gst_caps_append (caps, gst_caps_from_string (VIDEO_CAPS_STR))
```

Definition at line 38 of file gsttensor_converter_media_info_video.h.

9.29.2.2 is_video_supported

```
#define is_video_supported(
              ...  ) TRUE
```

Definition at line 41 of file gsttensor_converter_media_info_video.h.

9.29.2.3 NNS_VIDEO_FORMAT

```
#define NNS_VIDEO_FORMAT "{ RGB, BGR, RGBx, BGRx, xRGB, xBGR, RGBA, BGRA, ARGB, ABGR, GRAY8,
GRAY16_BE, GRAY16_LE }"
```

Caps string for supported video format.

Definition at line 31 of file gsttensor_converter_media_info_video.h.

9.29.2.4 VIDEO_CAPS_STR

```
#define VIDEO_CAPS_STR
```

Value:

```
    GST_VIDEO_CAPS_MAKE (NNS_VIDEO_FORMAT) \
    ", interlace-mode = (string) progressive"
```

Definition at line 34 of file gsttensor_converter_media_info_video.h.

9.30 nnstreamer/elements/gsttensor_converter_media_no_audio.h File Reference

Define collection of media type and functions to parse media info for audio if there is no audio support.

Macros

- #define append_audio_caps_template(caps)
- #define is_audio_supported(...) FALSE
- #define GstAudioInfo gsize
- #define gst_audio_info_init(i) memset (i, 0, sizeof (GstAudioInfo))
- #define gst_audio_info_from_caps(...) FALSE
- #define gst_audio_format_to_string(...) "Unknown"
- #define GST_AUDIO_INFO_FORMAT(...) GST_AUDIO_FORMAT_UNKNOWN
- #define GST_AUDIO_INFO_CHANNELS(...) 0
- #define GST_AUDIO_INFO_RATE(...) 0
- #define GST_AUDIO_INFO_BPF(...) 0

Enumerations

- enum GstAudioFormat {
  GST_AUDIO_FORMAT_UNKNOWN, GST_AUDIO_FORMAT_S8, GST_AUDIO_FORMAT_U8, GST_AUDIO_FORMAT_S16, GST_AUDIO_FORMAT_U16, GST_AUDIO_FORMAT_S32, GST_AUDIO_FORMAT_U32, GST_AUDIO_FORMAT_F32, GST_AUDIO_FORMAT_F64 }

9.30.1 Detailed Description

Define collection of media type and functions to parse media info for audio if there is no audio support.

NNStreamer media type definition for tensor-converter Copyright (C) 2019 Jijoong Moon jijoong.moon@samsung.com

Date

26 Mar 2019

See also

https://github.com/nnstreamer/nnstreamer

Author

Jijoong Moon jijoong.moon@samsung.com

Bug No known bugs except for NYI items

9.30.2 Macro Definition Documentation

9.30.2.1 append_audio_caps_template

```
#define append_audio_caps_template(
              caps )
```

Definition at line 23 of file gsttensor_converter_media_no_audio.h.

9.30.2.2 gst_audio_format_to_string

```
#define gst_audio_format_to_string(
              ... ) "Unknown"
```

Definition at line 42 of file gsttensor_converter_media_no_audio.h.

9.30.2.3 GST_AUDIO_INFO_BPF

```
#define GST_AUDIO_INFO_BPF(
              ... ) 0
```

Definition at line 47 of file gsttensor_converter_media_no_audio.h.

9.30.2.4 GST_AUDIO_INFO_CHANNELS

```
#define GST_AUDIO_INFO_CHANNELS(
              ... ) 0
```

Definition at line 45 of file gsttensor_converter_media_no_audio.h.

9.30.2.5 GST_AUDIO_INFO_FORMAT

```
#define GST_AUDIO_INFO_FORMAT(
              ... ) GST_AUDIO_FORMAT_UNKNOWN
```

Definition at line 44 of file gsttensor_converter_media_no_audio.h.

9.30.2.6 gst_audio_info_from_caps

```
#define gst_audio_info_from_caps(
              ... ) FALSE
```

Definition at line 41 of file gsttensor_converter_media_no_audio.h.

9.30.2.7 gst_audio_info_init

```
#define gst_audio_info_init(
              i ) memset (i, 0, sizeof (GstAudioInfo))
```

Definition at line 40 of file gsttensor_converter_media_no_audio.h.

9.30.2.8 GST_AUDIO_INFO_RATE

```
#define GST_AUDIO_INFO_RATE(
              ... ) 0
```

Definition at line 46 of file gsttensor_converter_media_no_audio.h.

9.30.2.9 GstAudioInfo

```
#define GstAudioInfo gsize
```

Definition at line 26 of file gsttensor_converter_media_no_audio.h.

9.30.2.10 is_audio_supported

```
#define is_audio_supported(
              ... ) FALSE
```

Definition at line 24 of file gsttensor_converter_media_no_audio.h.

9.30.3 Enumeration Type Documentation

9.30.3.1 GstAudioFormat

```
enum GstAudioFormat
```

Enumerator

GST_AUDIO_FORMAT_UNKNOWN	
GST_AUDIO_FORMAT_S8	
GST_AUDIO_FORMAT_U8	
GST_AUDIO_FORMAT_S16	
GST_AUDIO_FORMAT_U16	
GST_AUDIO_FORMAT_S32	
GST_AUDIO_FORMAT_U32	
GST_AUDIO_FORMAT_F32	
GST_AUDIO_FORMAT_F64	

Definition at line 28 of file gsttensor_converter_media_no_audio.h.

9.31 nnstreamer/elements/gsttensor_converter_media_no_video.h File Reference

Define collection of media type and functions to parse media info for video if there is no video support.

Macros

- #define append_video_caps_template(caps)
- #define is_video_supported(...) FALSE
- #define GstVideoInfo gsize
- #define gst_video_info_init(i) memset (i, 0, sizeof (GstVideoInfo))
- #define gst_video_info_from_caps(...) FALSE
- #define gst_video_format_to_string(...) "Unknown"
- #define GST_VIDEO_INFO_FORMAT(...) GST_VIDEO_FORMAT_UNKNOWN
- #define GST_VIDEO_INFO_WIDTH(...) 0
- #define GST_VIDEO_INFO_HEIGHT(...) 0
- #define GST_VIDEO_INFO_SIZE(...) 0
- #define GST_VIDEO_INFO_FPS_N(...) 0
- #define GST_VIDEO_INFO_FPS_D(...) 1

Enumerations

- enum GstVideoFormat {
  GST_VIDEO_FORMAT_UNKNOWN, GST_VIDEO_FORMAT_GRAY8, GST_VIDEO_FORMAT_RGB,
  GST_VIDEO_FORMAT_BGR,
  GST_VIDEO_FORMAT_RGBx, GST_VIDEO_FORMAT_BGRx, GST_VIDEO_FORMAT_xRGB, GST_VIDEO_FORMAT_xBG
  GST_VIDEO_FORMAT_RGBA, GST_VIDEO_FORMAT_BGRA, GST_VIDEO_FORMAT_ARGB, GST_VIDEO_FORMAT_AB
  GST_VIDEO_FORMAT_I420, GST_VIDEO_FORMAT_GRAY16_BE, GST_VIDEO_FORMAT_GRAY16_LE
  }

9.31.1 Detailed Description

Define collection of media type and functions to parse media info for video if there is no video support.

NNStreamer media type definition for tensor-converter Copyright (C) 2019 Jijoong Moon jijoong.moon@samsung.com

Date

26 Mar 2019

See also

https://github.com/nnstreamer/nnstreamer

Author

Jijoong Moon jijoong.moon@samsung.com

Bug No known bugs except for NYI items

9.31.2 Macro Definition Documentation

9.31.2.1 append_video_caps_template

```
#define append_video_caps_template(
              caps )
```

Definition at line 23 of file gsttensor_converter_media_no_video.h.

9.31.2.2 gst_video_format_to_string

```
#define gst_video_format_to_string(
              ... ) "Unknown"
```

Definition at line 48 of file gsttensor_converter_media_no_video.h.

9.31.2.3 GST_VIDEO_INFO_FORMAT

```
#define GST_VIDEO_INFO_FORMAT(
              ... ) GST_VIDEO_FORMAT_UNKNOWN
```

Definition at line 50 of file gsttensor_converter_media_no_video.h.

9.31.2.4 GST_VIDEO_INFO_FPS_D

```
#define GST_VIDEO_INFO_FPS_D(
              ... ) 1
```

Definition at line 55 of file gsttensor_converter_media_no_video.h.

9.31.2.5 GST_VIDEO_INFO_FPS_N

```
#define GST_VIDEO_INFO_FPS_N(
              ... ) 0
```

Definition at line 54 of file gsttensor_converter_media_no_video.h.

9.31.2.6 gst_video_info_from_caps

```
#define gst_video_info_from_caps(
              ...  ) FALSE
```

Definition at line 47 of file gsttensor_converter_media_no_video.h.

9.31.2.7 GST_VIDEO_INFO_HEIGHT

```
#define GST_VIDEO_INFO_HEIGHT(
              ...  ) 0
```

Definition at line 52 of file gsttensor_converter_media_no_video.h.

9.31.2.8 gst_video_info_init

```
#define gst_video_info_init(
              i ) memset (i, 0, sizeof (GstVideoInfo))
```

Definition at line 46 of file gsttensor_converter_media_no_video.h.

9.31.2.9 GST_VIDEO_INFO_SIZE

```
#define GST_VIDEO_INFO_SIZE(
              ...  ) 0
```

Definition at line 53 of file gsttensor_converter_media_no_video.h.

9.31.2.10 GST_VIDEO_INFO_WIDTH

```
#define GST_VIDEO_INFO_WIDTH(
              ...  ) 0
```

Definition at line 51 of file gsttensor_converter_media_no_video.h.

9.31.2.11 GstVideoInfo

```
#define GstVideoInfo gsize
```

Definition at line 26 of file gsttensor_converter_media_no_video.h.

9.31.2.12 is_video_supported

```
#define is_video_supported(
              ...  ) FALSE
```

Definition at line 24 of file gsttensor_converter_media_no_video.h.

9.31.3 Enumeration Type Documentation

9.31.3.1 GstVideoFormat

```
enum GstVideoFormat
```

Enumerator

GST_VIDEO_FORMAT_UNKNOWN	
GST_VIDEO_FORMAT_GRAY8	
GST_VIDEO_FORMAT_RGB	
GST_VIDEO_FORMAT_BGR	
GST_VIDEO_FORMAT_RGBx	
GST_VIDEO_FORMAT_BGRx	
GST_VIDEO_FORMAT_xRGB	
GST_VIDEO_FORMAT_xBGR	
GST_VIDEO_FORMAT_RGBA	
GST_VIDEO_FORMAT_BGRA	
GST_VIDEO_FORMAT_ARGB	
GST_VIDEO_FORMAT_ABGR	
GST_VIDEO_FORMAT_I420	
GST_VIDEO_FORMAT_GRAY16_BE	
GST_VIDEO_FORMAT_GRAY16_LE	

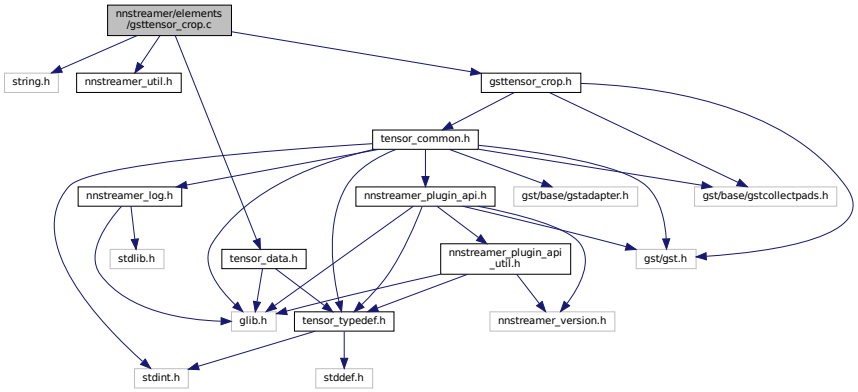
Definition at line 28 of file gsttensor_converter_media_no_video.h.

9.32 nnstreamer/elements/gsttensor_crop.c File Reference

GStreamer element to crop the regions of incoming tensor.

```
#include <string.h>
#include <nnstreamer_util.h>
#include "gsttensor_crop.h"
#include "tensor_data.h"
```

Include dependency graph for gsttensor_crop.c:



Classes

- struct tensor_region_s

  Internal data structure to describe tensor region.
- struct tensor_crop_info_s

  Internal data structure to describe cropping tensor data.

Macros

- #define GST_CAT_DEFAULT gst_tensor_crop_debug
- #define DEFAULT_SILENT TRUE

  Flag to print minimized log.
- #define DEFAULT_LATENESS (-1)

  Default value to compare timestamp of raw and info buffer, in milliseconds (-1 means no synchronization).
- #define gst_tensor_crop_parent_class parent_class

Enumerations

- enum { PROP_0, PROP_LATENESS, PROP_SILENT }

  tensor_crop properties

Functions

- GST_DEBUG_CATEGORY_STATIC (gst_tensor_crop_debug)
- G_DEFINE_TYPE (GstTensorCrop, gst_tensor_crop, GST_TYPE_ELEMENT)
- static void gst_tensor_crop_finalize (GObject *object)

  Function to finalize instance.
- static void gst_tensor_crop_set_property (GObject *object, guint prop_id, const GValue *value, GParamSpec *pspec)

  Setter for tensor_crop properties.
- static void gst_tensor_crop_get_property (GObject *object, guint prop_id, GValue *value, GParamSpec *pspec)

Getter for tensor_crop properties.

- static GstStateChangeReturn gst_tensor_crop_change_state (GstElement *element, GstStateChange transition)

  Handle state transition.
- static gboolean gst_tensor_crop_src_event (GstPad *pad, GstObject *parent, GstEvent *event)

  Handle event on src pad.
- static gboolean gst_tensor_crop_sink_event (GstCollectPads *pads, GstCollectData *data, GstEvent *event, gpointer user_data)

  Handle event on sink pad.
- static GstFlowReturn gst_tensor_crop_collected (GstCollectPads *pads, gpointer user_data)

  Chain function called when the buffer is available on all of the collect pads.
- static void gst_tensor_crop_class_init (GstTensorCropClass *klass)

  Initialize the tensor_crop's class.
- static void gst_tensor_crop_pad_reset (GstTensorCropPadData *cpad)

  Clear and reset old pad data.
- static void gst_tensor_crop_reset (GstTensorCrop *self)

  Clear and reset old data in tensor_crop.
- static void gst_tensor_crop_init (GstTensorCrop *self)

  Initialize tensor_crop element.
- static GstFlowReturn gst_tensor_crop_negotiate (GstTensorCrop *self)

  Set pad caps if not negotiated.
- static gboolean gst_tensor_crop_prepare_out_meta (GstTensorCrop *self, gpointer buffer, GstTensorMetaInfo *meta, GstTensorInfo *info, gboolean *is_flexible)

  Internal function to prepare output meta info.
- static gboolean gst_tensor_crop_get_crop_info (GstTensorCrop *self, GstBuffer *info, tensor_crop_info_s *cinfo)

  Internal function to parse buffer and fill crop info.
- static GstBuffer * gst_tensor_crop_do_cropping (GstTensorCrop *self, GstBuffer *raw, tensor_crop_info_s *cinfo)

  Internal function to crop incoming buffer.
- static GstFlowReturn gst_tensor_crop_chain (GstTensorCrop *self, GstCollectData *data_raw, GstCollectData *data_info)

  Internal function to transform the input buffer.

Variables

- static GstStaticPadTemplate raw_template

  Template for sink pad (raw data).
- static GstStaticPadTemplate info_template

  Template for sink pad (crop info).
- static GstStaticPadTemplate src_template

  Template for src pad.

9.32.1 Detailed Description

GStreamer element to crop the regions of incoming tensor.

Date

10 May 2021

See also

https://github.com/nnstreamer/nnstreamer

Author

Jaeyun Jung jy1210.jung@samsung.com

Bug No known bugs except for NYI items

9.32.2 Macro Definition Documentation

9.32.2.1 DEFAULT_LATENESS

```
#define DEFAULT_LATENESS (-1)
```

Default value to compare timestamp of raw and info buffer, in milliseconds (-1 means no synchronization).

Definition at line 89 of file gsttensor_crop.c.

9.32.2.2 DEFAULT_SILENT

```
#define DEFAULT_SILENT TRUE
```

Flag to print minimized log.

Definition at line 84 of file gsttensor_crop.c.

9.32.2.3 GST_CAT_DEFAULT

```
#define GST_CAT_DEFAULT gst_tensor_crop_debug
```

Definition at line 69 of file gsttensor_crop.c.

9.32.2.4 gst_tensor_crop_parent_class

```
#define gst_tensor_crop_parent_class parent_class
```

Definition at line 116 of file gsttensor_crop.c.

9.32.3 Enumeration Type Documentation

9.32.3.1 anonymous enum

```
anonymous enum
```

tensor_crop properties

Enumerator

PROP_0	
PROP_LATENESS	
PROP_SILENT	

Definition at line 74 of file gsttensor_crop.c.

9.32.4 Function Documentation

9.32.4.1 G_DEFINE_TYPE()

```
G_DEFINE_TYPE (
             GstTensorCrop ,
             gst_tensor_crop ,
             GST_TYPE_ELEMENT  )
```

9.32.4.2 GST_DEBUG_CATEGORY_STATIC()

```
GST_DEBUG_CATEGORY_STATIC (
             gst_tensor_crop_debug  )
```

9.32.4.3 gst_tensor_crop_chain()

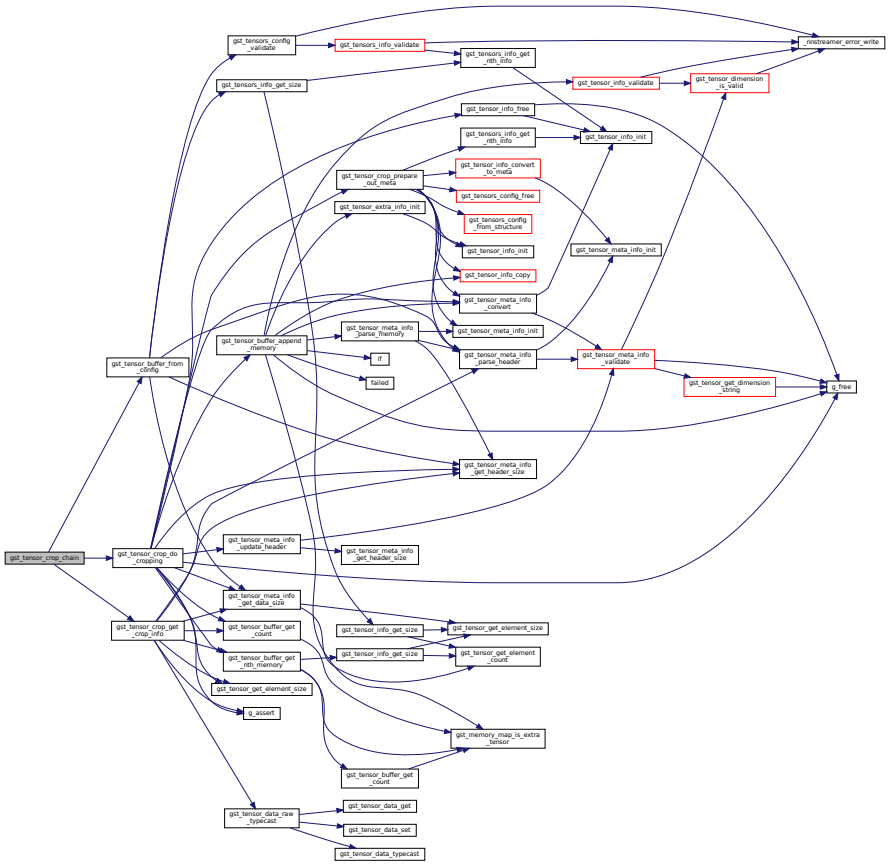
```
static GstFlowReturn gst_tensor_crop_chain (
             GstTensorCrop * self,
             GstCollectData * data_raw,
             GstCollectData * data_info )  [static]
```

Internal function to transform the input buffer.

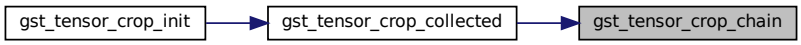
The case when raw and info have different timestamp. Compare timestamp and if time diff is less than lateness, crop raw buffer.

Definition at line 725 of file gsttensor_crop.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.32.4.4 gst_tensor_crop_change_state()

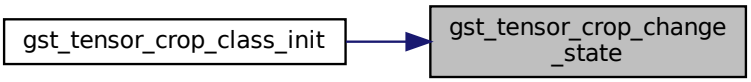
```
static GstStateChangeReturn gst_tensor_crop_change_state (
            GstElement * element,
            GstStateChange transition )  [static]
```

Handle state transition.

Definition at line 333 of file gsttensor_crop.c.

Here is the call graph for this function:

Here is the caller graph for this function:



9.32.4.5 gst_tensor_crop_class_init()

```
static void gst_tensor_crop_class_init (
            GstTensorCropClass * klass )  [static]
```

Initialize the tensor_crop's class.

GstTensorCrop::lateness:

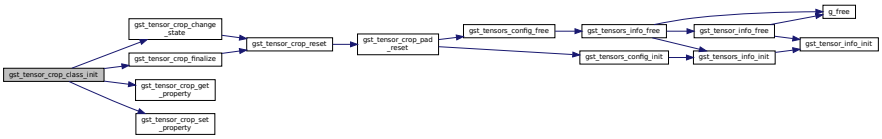
The time difference between raw and info buffer, in milliseconds (-1 means no synchronization). If raw and info buffers on the pads have different timestamp and time-diff is larger than 'lateness', tensor-crop will drop old buffer and wait for next buffers.

GstTensorCrop::silent:

The flag to enable/disable debugging messages.

Definition at line 137 of file gsttensor_crop.c.

Here is the call graph for this function:



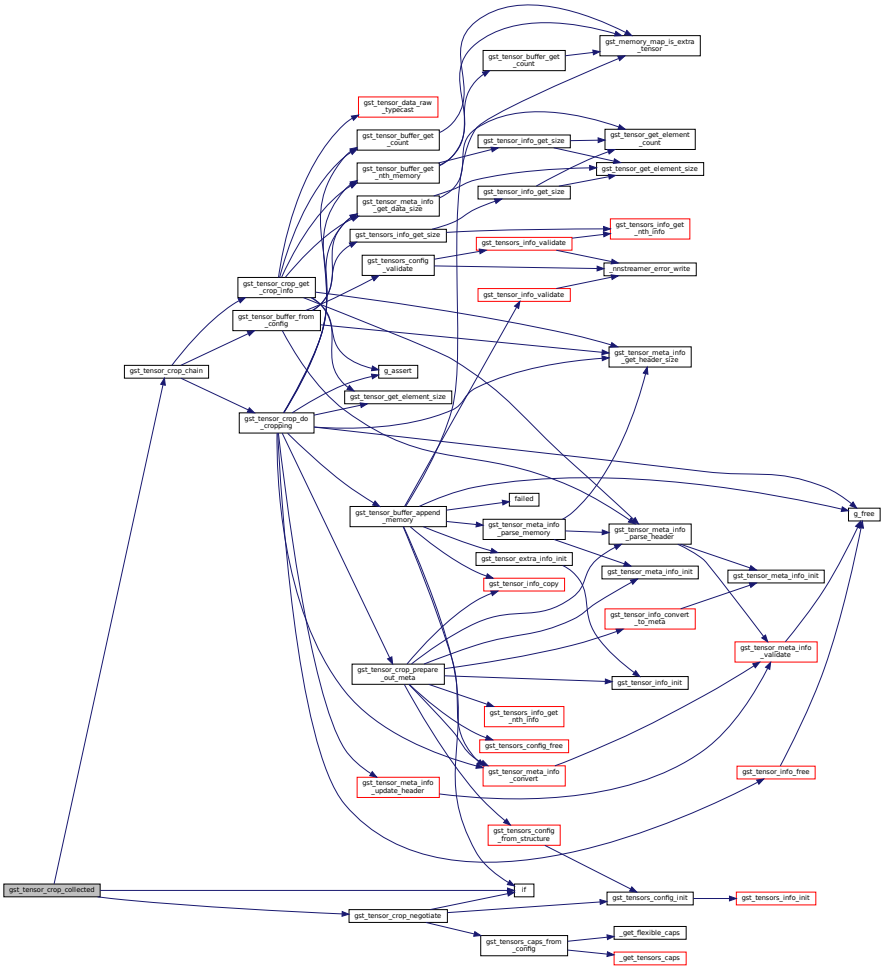
9.32.4.6 gst_tensor_crop_collected()

```
static GstFlowReturn gst_tensor_crop_collected (
            GstCollectPads * pads,
            gpointer user_data )  [static]
```

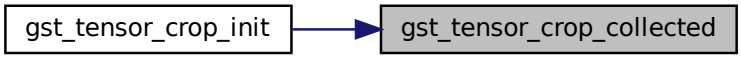
Chain function called when the buffer is available on all of the collect pads.

Definition at line 812 of file gsttensor_crop.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.32.4.7 gst_tensor_crop_do_cropping()

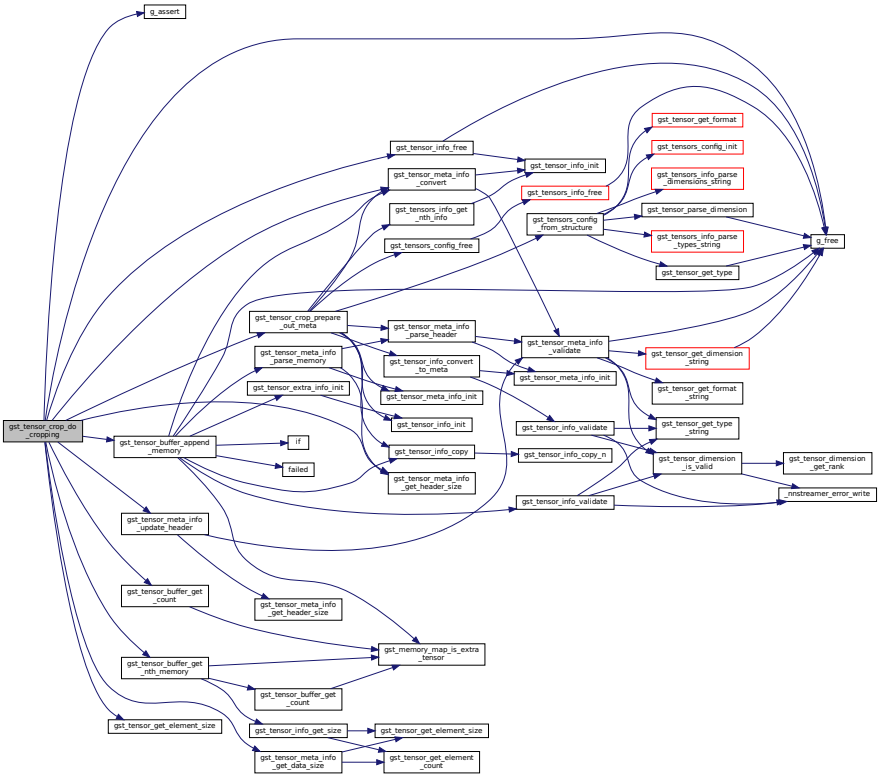
```
static GstBuffer* gst_tensor_crop_do_cropping (
            GstTensorCrop * self,
            GstBuffer * raw,
            tensor_crop_info_s * cinfo )  [static]
```

Internal function to crop incoming buffer.

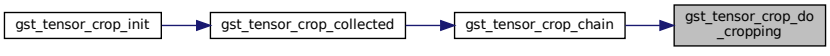
Todo Add various mode to crop tensor.

Definition at line 624 of file gsttensor_crop.c.

Here is the call graph for this function:



Here is the caller graph for this function:



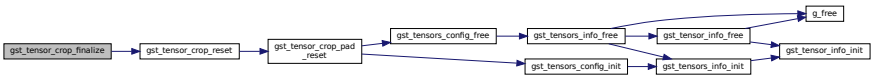
9.32.4.8 gst_tensor_crop_finalize()

```
static void gst_tensor_crop_finalize (
            GObject * object )  [static]
```

Function to finalize instance.

Definition at line 265 of file gsttensor_crop.c.

Here is the call graph for this function:



Here is the caller graph for this function:

9.32.4.9 gst_tensor_crop_get_crop_info()

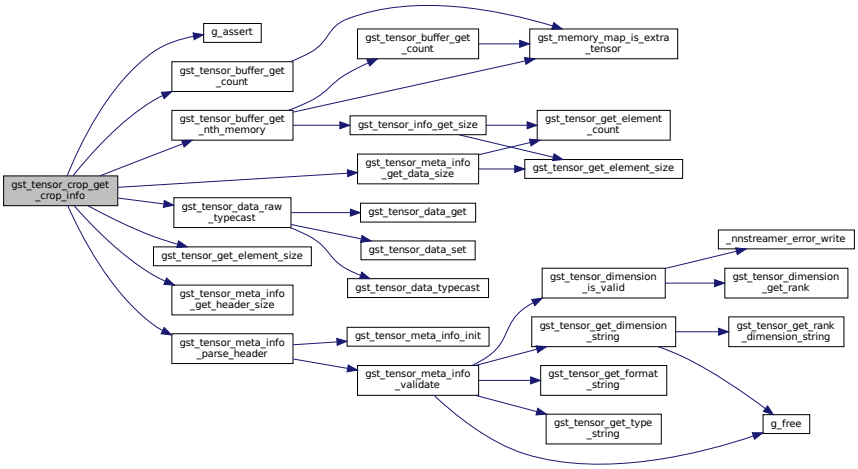
```
static gboolean gst_tensor_crop_get_crop_info (
            GstTensorCrop * self,
            GstBuffer * info,
            tensor_crop_info_s * cinfo )  [static]
```

Internal function to parse buffer and fill crop info.

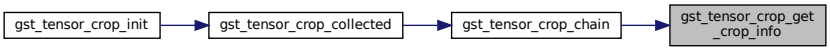
Todo Add various mode to crop tensor. Now tensor-crop handles NHWC data format only.

Definition at line 547 of file gsttensor_crop.c.

Here is the call graph for this function:



Here is the caller graph for this function:



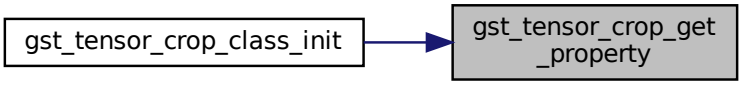
9.32.4.10 gst_tensor_crop_get_property()

```
static void gst_tensor_crop_get_property (
            GObject * object,
            guint prop_id,
            GValue * value,
            GParamSpec * pspec )  [static]
```

Getter for tensor_crop properties.

Definition at line 309 of file gsttensor_crop.c.

Here is the caller graph for this function:



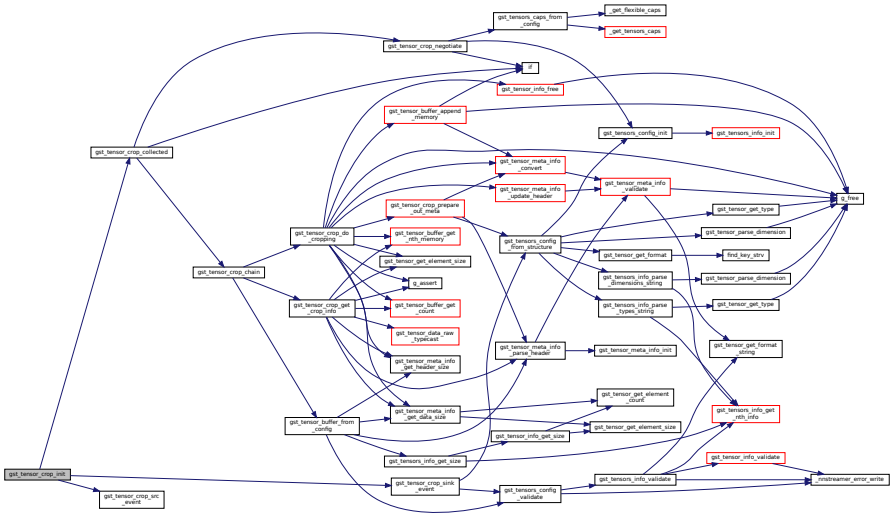
9.32.4.11 gst_tensor_crop_init()

```
static void gst_tensor_crop_init (
            GstTensorCrop * self )  [static]
```

Initialize tensor_crop element.

Definition at line 228 of file gsttensor_crop.c.

Here is the call graph for this function:



9.32.4.12 gst_tensor_crop_negotiate()

```
static GstFlowReturn gst_tensor_crop_negotiate (
            GstTensorCrop * self )  [static]
```

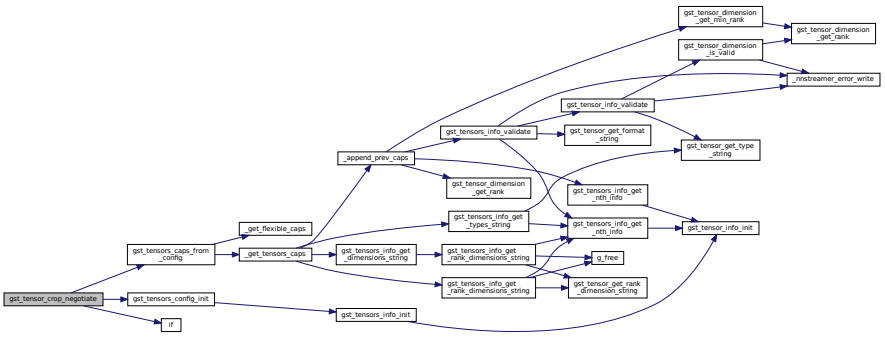
Set pad caps if not negotiated.

Cannot use gst-pad util to get stream ID (multiple sink pads). Create stream ID using first sink pad.

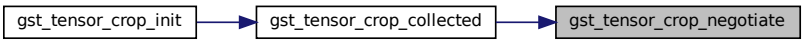
Get config from collect-pads and set framerate. Output is always flexible tensor.

Definition at line 423 of file gsttensor_crop.c.

Here is the call graph for this function:



Here is the caller graph for this function:



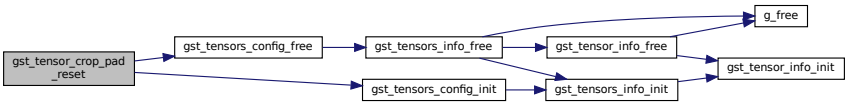
9.32.4.13 gst_tensor_crop_pad_reset()

```
static void gst_tensor_crop_pad_reset (
            GstTensorCropPadData * cpad )  [static]
```

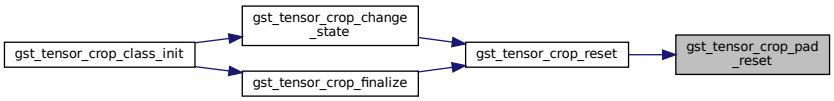
Clear and reset old pad data.

Definition at line 195 of file gsttensor_crop.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.32.4.14 gst_tensor_crop_prepare_out_meta()

```
static gboolean gst_tensor_crop_prepare_out_meta (
            GstTensorCrop * self,
            gpointer buffer,
            GstTensorMetaInfo * meta,
            GstTensorInfo * info,
            gboolean * is_flexible )  [static]
```

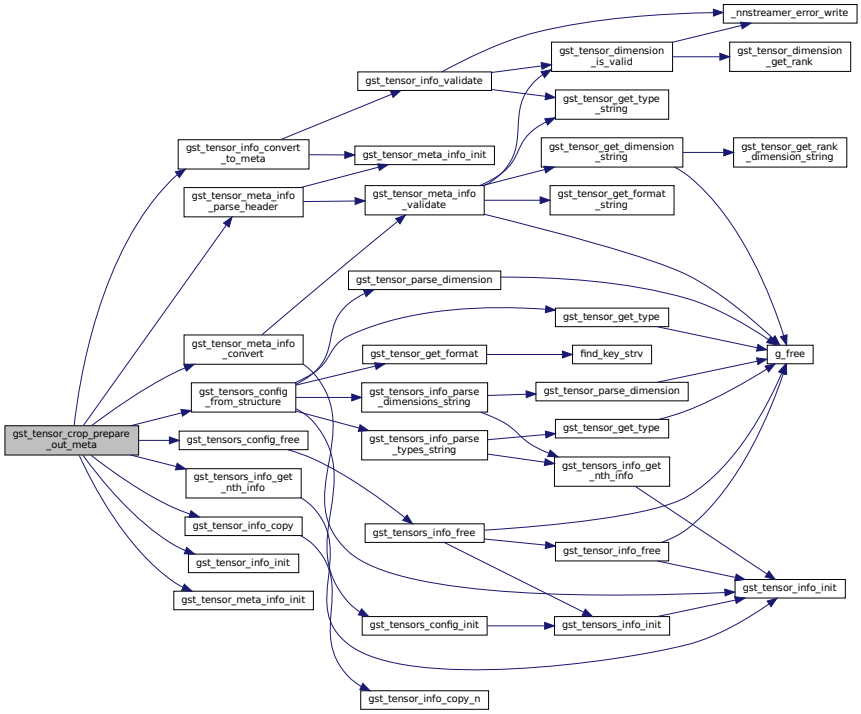
Internal function to prepare output meta info.

Note

tensor-crop handles single tensor. Parse first one.

Definition at line 497 of file gsttensor_crop.c.

Here is the call graph for this function:



Here is the caller graph for this function:

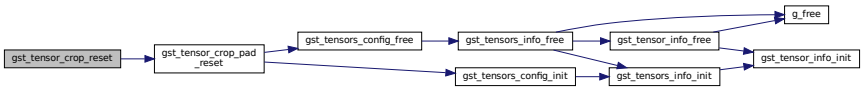
9.32.4.15 gst_tensor_crop_reset()

```
static void gst_tensor_crop_reset (
            GstTensorCrop * self )  [static]
```

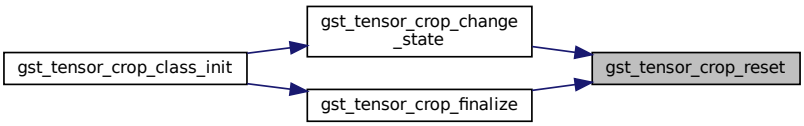
Clear and reset old data in tensor_crop.

Definition at line 205 of file gsttensor_crop.c.

Here is the call graph for this function:



Here is the caller graph for this function:



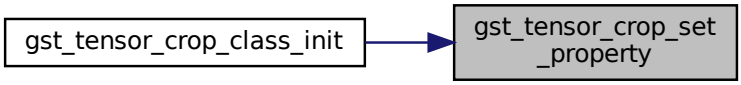
9.32.4.16 gst_tensor_crop_set_property()

```
static void gst_tensor_crop_set_property (
            GObject * object,
            guint prop_id,
            const GValue * value,
            GParamSpec * pspec )  [static]
```

Setter for tensor_crop properties.

Definition at line 285 of file gsttensor_crop.c.

Here is the caller graph for this function:



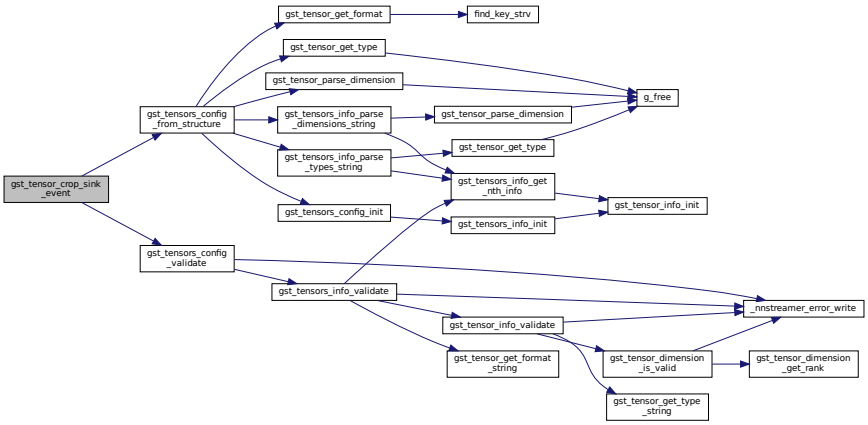
9.32.4.17 gst_tensor_crop_sink_event()

```
static gboolean gst_tensor_crop_sink_event (
            GstCollectPads * pads,
            GstCollectData * data,
            GstEvent * event,
            gpointer user_data )  [static]
```

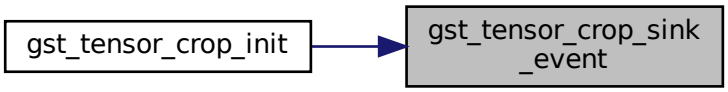
Handle event on sink pad.

Definition at line 388 of file gsttensor_crop.c.

Here is the call graph for this function:



Here is the caller graph for this function:



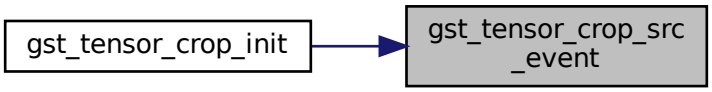
9.32.4.18 gst_tensor_crop_src_event()

```
static gboolean gst_tensor_crop_src_event (
            GstPad * pad,
            GstObject * parent,
            GstEvent * event )  [static]
```

Handle event on src pad.

Definition at line 368 of file gsttensor_crop.c.

Here is the caller graph for this function:



9.32.5 Variable Documentation

9.32.5.1 info_template

```
GstStaticPadTemplate info_template  [static]
```

Initial value:

```
= GST_STATIC_PAD_TEMPLATE ("info",
    GST_PAD_SINK,
    GST_PAD_ALWAYS,
    GST_STATIC_CAPS (GST_TENSORS_FLEX_CAP_DEFAULT))
```

Template for sink pad (crop info).

Definition at line 103 of file gsttensor_crop.c.

9.32.5.2 raw_template

```
GstStaticPadTemplate raw_template  [static]
```

Initial value:

```
= GST_STATIC_PAD_TEMPLATE ("raw",
    GST_PAD_SINK,
    GST_PAD_ALWAYS,
    GST_STATIC_CAPS (GST_TENSOR_CAP_DEFAULT ";"
        GST_TENSORS_CAP_MAKE ("{ static, flexible }")))
```

Template for sink pad (raw data).

Definition at line 94 of file gsttensor_crop.c.

9.32.5.3 src_template

```
GstStaticPadTemplate src_template  [static]
```

Initial value:

```
= GST_STATIC_PAD_TEMPLATE ("src",
    GST_PAD_SRC,
    GST_PAD_ALWAYS,
    GST_STATIC_CAPS (GST_TENSORS_FLEX_CAP_DEFAULT))
```

Template for src pad.

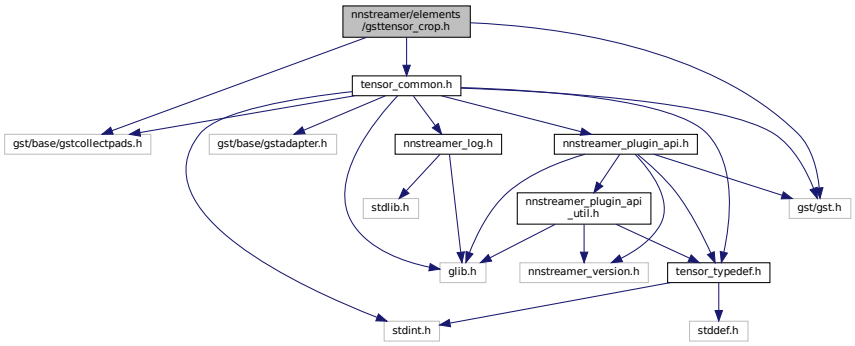
Definition at line 111 of file gsttensor_crop.c.

9.33 nnstreamer/elements/gsttensor_crop.h File Reference

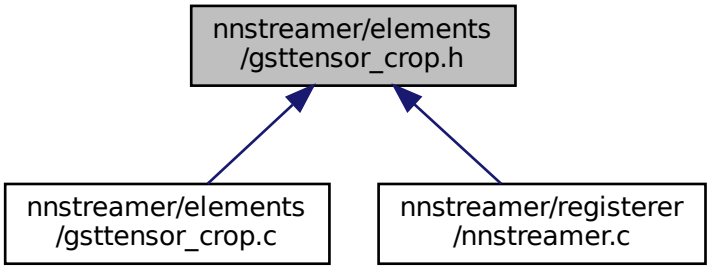
GStreamer element to crop the regions of incoming tensor.

```
#include <gst/gst.h>
#include <gst/base/gstcollectpads.h>
#include <tensor_common.h>
```

Include dependency graph for gsttensor_crop.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct GstTensorCropPadData

  GstTensorCrop pad data.
- struct _GstTensorCrop

  GstTensorCrop data structure.
- struct _GstTensorCropClass

  GstTensorCropClass data structure.

Macros

- #define GST_TYPE_TENSOR_CROP (gst_tensor_crop_get_type())
- #define GST_TENSOR_CROP(obj) (G_TYPE_CHECK_INSTANCE_CAST((obj),GST_TYPE_TENSOR_CROP,GstTensorCrop
- #define GST_TENSOR_CROP_CLASS(klass) (G_TYPE_CHECK_CLASS_CAST((klass),GST_TYPE_TENSOR_CROP,GstTe
- #define GST_IS_TENSOR_CROP(obj) (G_TYPE_CHECK_INSTANCE_TYPE((obj),GST_TYPE_TENSOR_CROP))
- #define GST_IS_TENSOR_CROP_CLASS(klass) (G_TYPE_CHECK_CLASS_TYPE((klass),GST_TYPE_TENSOR_CROP))

Typedefs

- typedef struct _GstTensorCrop GstTensorCrop
- typedef struct _GstTensorCropClass GstTensorCropClass

Functions

- GType gst_tensor_crop_get_type (void)

  Function to get type of tensor_crop.

9.33.1 Detailed Description

GStreamer element to crop the regions of incoming tensor.

Date

10 May 2021

See also

https://github.com/nnstreamer/nnstreamer

Author

Jaeyun Jung jy1210.jung@samsung.com

Bug No known bugs except for NYI items

9.33.2 Macro Definition Documentation

9.33.2.1 GST_IS_TENSOR_CROP

```
#define GST_IS_TENSOR_CROP(
              obj ) (G_TYPE_CHECK_INSTANCE_TYPE((obj),GST_TYPE_TENSOR_CROP))
```

Definition at line 28 of file gsttensor_crop.h.

9.33.2.2 GST_IS_TENSOR_CROP_CLASS

```
#define GST_IS_TENSOR_CROP_CLASS(
              klass ) (G_TYPE_CHECK_CLASS_TYPE((klass),GST_TYPE_TENSOR_CROP))
```

Definition at line 30 of file gsttensor_crop.h.

9.33.2.3 GST_TENSOR_CROP

```
#define GST_TENSOR_CROP(
              obj ) (G_TYPE_CHECK_INSTANCE_CAST((obj),GST_TYPE_TENSOR_CROP,GstTensorCrop))
```

Definition at line 24 of file gsttensor_crop.h.

9.33.2.4 GST_TENSOR_CROP_CLASS

```
#define GST_TENSOR_CROP_CLASS(
              klass ) (G_TYPE_CHECK_CLASS_CAST((klass),GST_TYPE_TENSOR_CROP,GstTensorCropClass))
```

Definition at line 26 of file gsttensor_crop.h.

9.33.2.5 GST_TYPE_TENSOR_CROP

```
#define GST_TYPE_TENSOR_CROP (gst_tensor_crop_get_type())
```

Definition at line 22 of file gsttensor_crop.h.

9.33.3 Typedef Documentation

9.33.3.1 GstTensorCrop

```
typedef struct _GstTensorCrop GstTensorCrop
```

Definition at line 33 of file gsttensor_crop.h.

9.33.3.2 GstTensorCropClass

```
typedef struct _GstTensorCropClass GstTensorCropClass
```

Definition at line 34 of file gsttensor_crop.h.

9.33.4 Function Documentation

9.33.4.1 gst_tensor_crop_get_type()

```
GType gst_tensor_crop_get_type (
            void  )
```

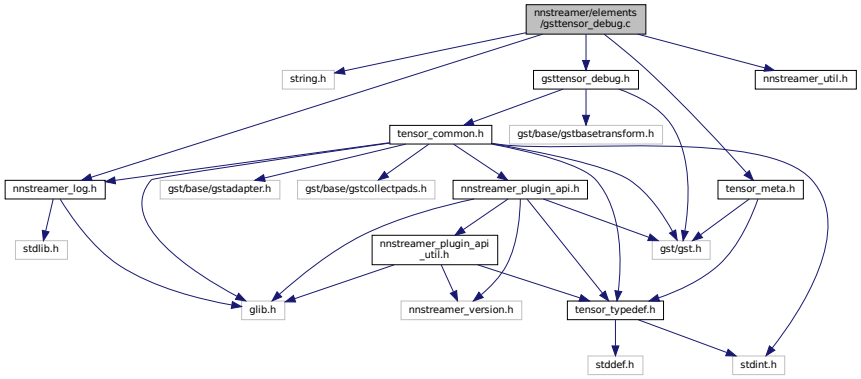
Function to get type of tensor_crop.

9.34 nnstreamer/elements/gsttensor_debug.c File Reference

GStreamer plugin to help debug tensor streams.

```
#include <string.h>
#include <nnstreamer_log.h>
#include <nnstreamer_util.h>
#include "gsttensor_debug.h"
#include "tensor_meta.h"
```

Include dependency graph for gsttensor_debug.c:



Macros

- #define DBG (!self->silent)

  Macro for debug mode.
- #define GST_CAT_DEFAULT gst_tensor_debug_debug
- #define CAPS_STRING GST_TENSORS_CAP_MAKE(GST_TENSOR_FORMAT_ALL)
- #define C_FLAGS(v) ((guint) v)
- #define TENSOR_DEBUG_TYPE_OUTPUT_FLAGS (tensor_debug_output_flags_get_type())
- #define DEFAULT_TENSOR_DEBUG_OUTPUT_FLAGS (TDBG_OUTPUT_CONSOLE_I)
- #define TENSOR_DEBUG_TYPE_CAPS (tensor_debug_cap_get_type())
- #define DEFAULT_TENSOR_DEBUG_CAP (TDBG_CAP_SHOW_UPDATE_F)
- #define TENSOR_DEBUG_TYPE_META_FLAGS (tensor_debug_meta_flags_get_type())
- #define DEFAULT_TENSOR_DEBUG_META_FLAGS (TDBG_META_DISABLED)
- #define DEFAULT_SILENT TRUE

  Flag to print minimized log.
- #define gst_tensor_debug_parent_class parent_class

Enumerations

- enum {
  PROP_0, PROP_SILENT, PROP_OUTPUT, PROP_CAP,
  PROP_META }

  tensor_debug properties

Functions

- GST_DEBUG_CATEGORY_STATIC (gst_tensor_debug_debug)
- static GType tensor_debug_output_flags_get_type (void)

  Flags for output_mode of GstTensorDebug.
- static GType tensor_debug_cap_get_type (void)

  Enums for cap_mode of GstTensorDebug.
- static GType tensor_debug_meta_flags_get_type (void)

  Flags for meta_mode of GstTensorDebug.
- G_DEFINE_TYPE (GstTensorDebug, gst_tensor_debug, GST_TYPE_BASE_TRANSFORM)
- static void gst_tensor_debug_set_property (GObject *object, guint prop_id, const GValue *value, GParamSpec *pspec)

  Setter for tensor_debug properties.
- static void gst_tensor_debug_get_property (GObject *object, guint prop_id, GValue *value, GParamSpec *pspec)

  Getter for tensor_debug properties.
- static void gst_tensor_debug_finalize (GObject *object)

  Function to finalize instance.
- static GstFlowReturn gst_tensor_debug_transform_ip (GstBaseTransform *trans, GstBuffer *buffer)

  in-place transform
- static GstCaps * gst_tensor_debug_fixate_caps (GstBaseTransform *trans, GstPadDirection direction, GstCaps *caps, GstCaps *othercaps)

  fixate caps. required vmethod of GstBaseTransform.
- static gboolean gst_tensor_debug_set_caps (GstBaseTransform *trans, GstCaps *in_caps, GstCaps *out_caps)

  set caps. required vmethod of GstBaseTransform.
- static void gst_tensor_debug_class_init (GstTensorDebugClass *klass)

  Initialize the tensor_debug's class.
- static void gst_tensor_debug_init (GstTensorDebug *self)

  Initialize tensor_debug element.
- static void _gst_tensor_debug_output (GstTensorDebug *self, GstBuffer *buffer)

  The core function that provides debug output based on the contents.

Variables

- static GstStaticPadTemplate sink_factory

  The capabilities of the inputs.
- static GstStaticPadTemplate src_factory

  The capabilities of the outputs.

9.34.1 Detailed Description

GStreamer plugin to help debug tensor streams.

GStreamer/NNStreamer tensor_debug Copyright (C) 2022 MyungJoo Ham myungjoo.ham@samsung.com

Date

23 Sep 2022

See also

https://github.com/nnstreamer/nnstreamer

Author

MyungJoo Ham myungjoo.ham@samsung.com

Bug No known bugs except for NYI items

9.34.2 Macro Definition Documentation

9.34.2.1 C_FLAGS

```
#define C_FLAGS(
              v ) ((guint) v)
```

Definition at line 88 of file gsttensor_debug.c.

9.34.2.2 CAPS_STRING

```
#define CAPS_STRING GST_TENSORS_CAP_MAKE(GST_TENSOR_FORMAT_ALL)
```

This is a new element created after the obsoletion of other/tensor. Use other/tensors if you want to use tensor_debug

Definition at line 58 of file gsttensor_debug.c.

9.34.2.3 DBG

```
#define DBG (!self->silent)
```

Macro for debug mode.

SECTION:element-tensor_debug

A filter that generates debug messages for developer at the insertion point of the given pipeline. An application writer using an nnstreamer pipeline can use tensor_debug to debug or get profile information in their applications.

Note that this does not support other/tensor, but only supports other/tensors.

<refsect2> <title>Example launch line</title> |[gst-launch-1.0 videotestsrc ! video/x-raw,format=RGB,width=640,height=480 ! tensor_converter ! tensor_debug output-method=console-info capability=always ! tensor_sink]| </refsect2>

Definition at line 48 of file gsttensor_debug.c.

9.34.2.4 DEFAULT_SILENT

```
#define DEFAULT_SILENT TRUE
```

Flag to print minimized log.

Definition at line 191 of file gsttensor_debug.c.

9.34.2.5 DEFAULT_TENSOR_DEBUG_CAP

```
#define DEFAULT_TENSOR_DEBUG_CAP (TDBG_CAP_SHOW_UPDATE_F)
```

Definition at line 161 of file gsttensor_debug.c.

9.34.2.6 DEFAULT_TENSOR_DEBUG_META_FLAGS

```
#define DEFAULT_TENSOR_DEBUG_META_FLAGS (TDBG_META_DISABLED)
```

Definition at line 186 of file gsttensor_debug.c.

9.34.2.7 DEFAULT_TENSOR_DEBUG_OUTPUT_FLAGS

```
#define DEFAULT_TENSOR_DEBUG_OUTPUT_FLAGS (TDBG_OUTPUT_CONSOLE_I)
```

Definition at line 135 of file gsttensor_debug.c.

9.34.2.8 GST_CAT_DEFAULT

```
#define GST_CAT_DEFAULT gst_tensor_debug_debug
```

Definition at line 52 of file gsttensor_debug.c.

9.34.2.9 gst_tensor_debug_parent_class

```
#define gst_tensor_debug_parent_class parent_class
```

Definition at line 193 of file gsttensor_debug.c.

9.34.2.10 TENSOR_DEBUG_TYPE_CAPS

```
#define TENSOR_DEBUG_TYPE_CAPS (tensor_debug_cap_get_type())
```

Definition at line 137 of file gsttensor_debug.c.

9.34.2.11 TENSOR_DEBUG_TYPE_META_FLAGS

```
#define TENSOR_DEBUG_TYPE_META_FLAGS (tensor_debug_meta_flags_get_type())
```

Definition at line 163 of file gsttensor_debug.c.

9.34.2.12 TENSOR_DEBUG_TYPE_OUTPUT_FLAGS

```
#define TENSOR_DEBUG_TYPE_OUTPUT_FLAGS (tensor_debug_output_flags_get_type())
```

Definition at line 90 of file gsttensor_debug.c.

9.34.3 Enumeration Type Documentation

9.34.3.1 anonymous enum

```
anonymous enum
```

tensor_debug properties

Enumerator

PROP_0	
PROP_SILENT	
PROP_OUTPUT	
PROP_CAP	
PROP_META	

Definition at line 79 of file gsttensor_debug.c.

9.34.4 Function Documentation

9.34.4.1 _gst_tensor_debug_output()

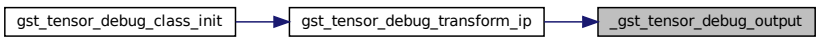
```
static void _gst_tensor_debug_output (
            GstTensorDebug * self,
            GstBuffer * buffer )  [static]
```

The core function that provides debug output based on the contents.

Todo NYI: do the debug task

Definition at line 397 of file gsttensor_debug.c.

Here is the caller graph for this function:



9.34.4.2 G_DEFINE_TYPE()

```
G_DEFINE_TYPE (
            GstTensorDebug ,
            gst_tensor_debug ,
            GST_TYPE_BASE_TRANSFORM  )
```

9.34.4.3 GST_DEBUG_CATEGORY_STATIC()

```
GST_DEBUG_CATEGORY_STATIC (
            gst_tensor_debug_debug  )
```

9.34.4.4 gst_tensor_debug_class_init()

```
static void gst_tensor_debug_class_init (
            GstTensorDebugClass * klass )  [static]
```

Initialize the tensor_debug's class.

GstTensorDebug::silent:

The flag to enable/disable debugging messages.

GstTensorDebug::output:

The combination of enums configuring output methods.

Todo check the behavior of name and nick (output methods vs output)

GstTensorDebug::cap:

The logging preference of the stream capability (GSTCAP).

GstTensorDebug::meta:

The logging preference of in-stream metadata (GSTMETA).

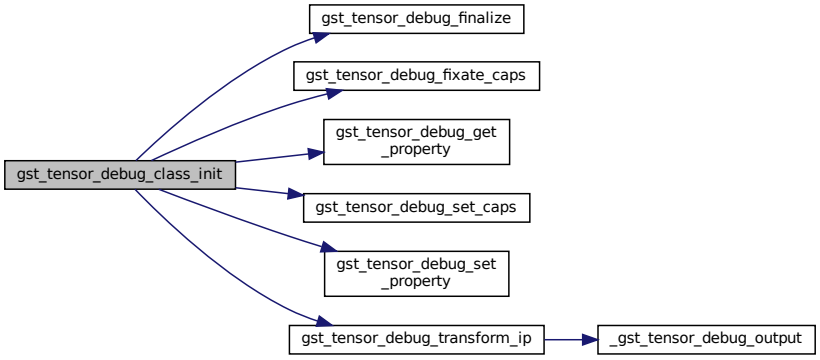
This won't modify the contents!

call transform_ip although it's passthrough

Note. Without transform_caps and with passthrough_on_same_caps = TRUE, This element is not allowed to touch the contents, but can inspect the contents with transform_ip by setting transform_ip_on_passthrough.

Definition at line 215 of file gsttensor_debug.c.

Here is the call graph for this function:



9.34.4.5 gst_tensor_debug_finalize()

```
static void gst_tensor_debug_finalize (
            GObject * object )  [static]
```

Function to finalize instance.

Definition at line 327 of file gsttensor_debug.c.

Here is the caller graph for this function:

9.34.4.6 gst_tensor_debug_fixate_caps()

```
static GstCaps * gst_tensor_debug_fixate_caps (
            GstBaseTransform * trans,
            GstPadDirection direction,
            GstCaps * caps,
            GstCaps * othercaps )  [static]
```

fixate caps. required vmethod of GstBaseTransform.

Definition at line 421 of file gsttensor_debug.c.

Here is the caller graph for this function:

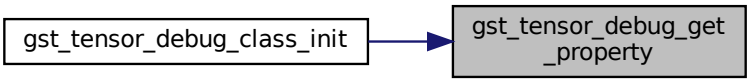
9.34.4.7 gst_tensor_debug_get_property()

```
static void gst_tensor_debug_get_property (
            GObject * object,
            guint prop_id,
            GValue * value,
            GParamSpec * pspec )  [static]
```

Getter for tensor_debug properties.

Definition at line 368 of file gsttensor_debug.c.

Here is the caller graph for this function:



9.34.4.8 gst_tensor_debug_init()

```
static void gst_tensor_debug_init (
            GstTensorDebug * self )  [static]
```

Initialize tensor_debug element.

init properties

Definition at line 313 of file gsttensor_debug.c.

9.34.4.9 gst_tensor_debug_set_caps()

```
static gboolean gst_tensor_debug_set_caps (
            GstBaseTransform * trans,
            GstCaps * incaps,
            GstCaps * outcaps )  [static]
```

set caps. required vmethod of GstBaseTransform.

Definition at line 435 of file gsttensor_debug.c.

Here is the caller graph for this function:

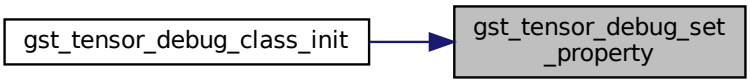
9.34.4.10 gst_tensor_debug_set_property()

```
static void gst_tensor_debug_set_property (
            GObject * object,
            guint prop_id,
            const GValue * value,
            GParamSpec * pspec )  [static]
```

Setter for tensor_debug properties.

Definition at line 336 of file gsttensor_debug.c.

Here is the caller graph for this function:



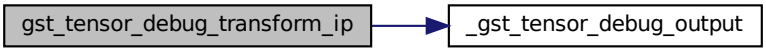
9.34.4.11 gst_tensor_debug_transform_ip()

```
static GstFlowReturn gst_tensor_debug_transform_ip (
            GstBaseTransform * trans,
            GstBuffer * buffer )  [static]
```

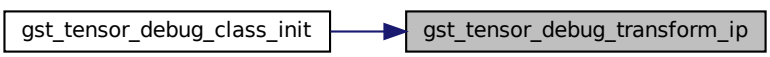
in-place transform

Definition at line 408 of file gsttensor_debug.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.34.4.12 tensor_debug_cap_get_type()

```
static GType tensor_debug_cap_get_type (
            void  )  [static]
```

Enums for cap_mode of GstTensorDebug.

Definition at line 142 of file gsttensor_debug.c.

9.34.4.13 tensor_debug_meta_flags_get_type()

```
static GType tensor_debug_meta_flags_get_type (
            void  )  [static]
```

Flags for meta_mode of GstTensorDebug.

Definition at line 168 of file gsttensor_debug.c.

9.34.4.14 tensor_debug_output_flags_get_type()

```
static GType tensor_debug_output_flags_get_type (
            void  )  [static]
```

Flags for output_mode of GstTensorDebug.

Definition at line 95 of file gsttensor_debug.c.

9.34.5 Variable Documentation

9.34.5.1 sink_factory

```
GstStaticPadTemplate sink_factory  [static]
```

Initial value:

```
= GST_STATIC_PAD_TEMPLATE ("sink",
    GST_PAD_SINK,
    GST_PAD_ALWAYS,
    GST_STATIC_CAPS (CAPS_STRING))
```

The capabilities of the inputs.

Definition at line 63 of file gsttensor_debug.c.

9.34.5.2 src_factory

```
GstStaticPadTemplate src_factory  [static]
```

Initial value:

```
= GST_STATIC_PAD_TEMPLATE ("src",
    GST_PAD_SRC,
    GST_PAD_ALWAYS,
    GST_STATIC_CAPS (CAPS_STRING))
```

The capabilities of the outputs.

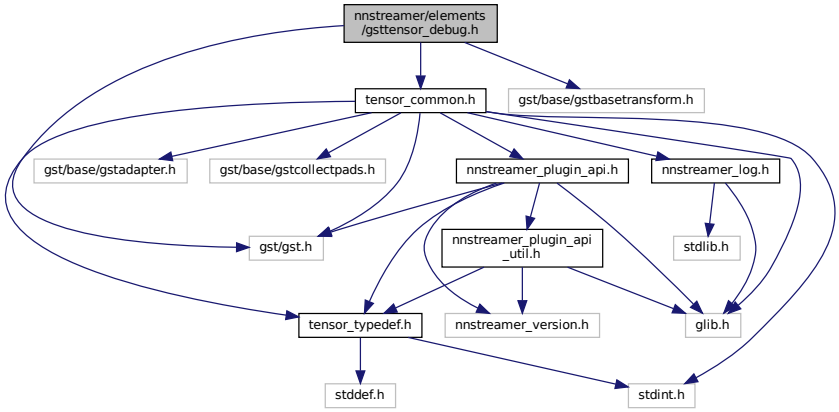
Definition at line 71 of file gsttensor_debug.c.

9.35 nnstreamer/elements/gsttensor_debug.h File Reference

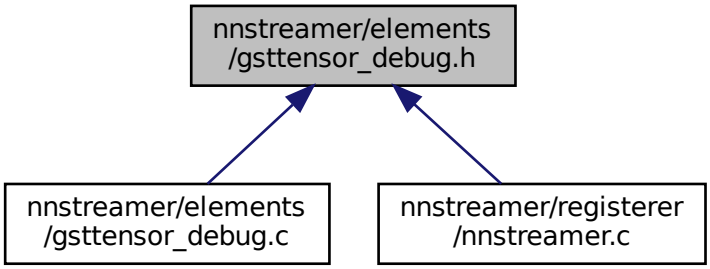
GStreamer plugin to help debug tensor streams.

```
#include <gst/gst.h>
#include <gst/base/gstbasetransform.h>
#include <tensor_common.h>
```

Include dependency graph for gsttensor_debug.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct _GstTensorDebug

  Internal data structure for tensor_debug instances.
- struct _GstTensorDebugClass

  GstTensorDebugClass data structure.

Macros

- #define GST_TYPE_TENSOR_DEBUG (gst_tensor_debug_get_type())
- #define GST_TENSOR_DEBUG(obj) (G_TYPE_CHECK_INSTANCE_CAST((obj),GST_TYPE_TENSOR_DEBUG,GstTensorD
- #define GST_TENSOR_DEBUG_CLASS(klass) (G_TYPE_CHECK_CLASS_CAST((klass),GST_TYPE_TENSOR_DEBUG,Gs
- #define GST_IS_TENSOR_DEBUG(obj) (G_TYPE_CHECK_INSTANCE_TYPE((obj),GST_TYPE_TENSOR_DEBUG))
- #define GST_IS_TENSOR_DEBUG_CLASS(klass) (G_TYPE_CHECK_CLASS_TYPE((klass),GST_TYPE_TENSOR_DEBUG
- #define GST_TENSOR_DEBUG_CAST(obj) ((GstTensorDebug *)(obj))

Typedefs

- typedef struct _GstTensorDebug GstTensorDebug
- typedef struct _GstTensorDebugClass GstTensorDebugClass

Enumerations

- enum tdbg_output_mode {
  TDBG_OUTPUT_DISABLED = 0x00, TDBG_OUTPUT_CONSOLE_I = 0x11, TDBG_OUTPUT_CONSOLE_W = 0x12, TDBG_OUTPUT_CONSOLE_E = 0x13,
  TDBG_OUTPUT_GSTDBG_I = 0x24, TDBG_OUTPUT_GSTDBG_W = 0x28, TDBG_OUTPUT_GSTDBG_E = 0x2C, TDBG_OUTPUT_CIRCULARBUF = 0x100,
  TDBG_OUTPUT_FILEWRITE = 0x200 }

  Property "OUTPUT" specification.
- enum tdbg_cap_mode { TDBG_CAP_DISABLED = 0, TDBG_CAP_SHOW_UPDATE = 1, TDBG_CAP_SHOW_UPDATE_F = 2, TDBG_CAP_SHOW_ALWAYS = 3 }

  Property "CAP" specification.
- enum tdbg_meta_mode { TDBG_META_DISABLED = 0x0, TDBG_META_TIMESTAMP = 0x1, TDBG_META_QUERYSERVER = 0x2 }

  Property "META" specification.

Functions

- GType gst_tensor_debug_get_type (void)

  Get Type function required for gst elements.

9.35.1 Detailed Description

GStreamer plugin to help debug tensor streams.

GStreamer/NNStreamer tensor_debug Copyright (C) 2022 MyungJoo Ham myungjoo.ham@samsung.com

Date

23 Sep 2022

See also

https://github.com/nnstreamer/nnstreamer

Author

MyungJoo Ham myungjoo.ham@samsung.com

Bug No known bugs except for NYI items

9.35.2 Macro Definition Documentation

9.35.2.1 GST_IS_TENSOR_DEBUG

```
#define GST_IS_TENSOR_DEBUG(
              obj ) (G_TYPE_CHECK_INSTANCE_TYPE((obj),GST_TYPE_TENSOR_DEBUG))
```

Definition at line 35 of file gsttensor_debug.h.

9.35.2.2 GST_IS_TENSOR_DEBUG_CLASS

```
#define GST_IS_TENSOR_DEBUG_CLASS(
              klass ) (G_TYPE_CHECK_CLASS_TYPE((klass),GST_TYPE_TENSOR_DEBUG))
```

Definition at line 37 of file gsttensor_debug.h.

9.35.2.3 GST_TENSOR_DEBUG

```
#define GST_TENSOR_DEBUG(
              obj ) (G_TYPE_CHECK_INSTANCE_CAST((obj),GST_TYPE_TENSOR_DEBUG,GstTensorDebug))
```

Definition at line 31 of file gsttensor_debug.h.

9.35.2.4 GST_TENSOR_DEBUG_CAST

```
#define GST_TENSOR_DEBUG_CAST(
              obj ) ((GstTensorDebug *)(obj))
```

Definition at line 39 of file gsttensor_debug.h.

9.35.2.5 GST_TENSOR_DEBUG_CLASS

```
#define GST_TENSOR_DEBUG_CLASS(
              klass ) (G_TYPE_CHECK_CLASS_CAST((klass),GST_TYPE_TENSOR_DEBUG,GstTensorDebugClass))
```

Definition at line 33 of file gsttensor_debug.h.

9.35.2.6 GST_TYPE_TENSOR_DEBUG

```
#define GST_TYPE_TENSOR_DEBUG (gst_tensor_debug_get_type())
```

Definition at line 29 of file gsttensor_debug.h.

9.35.3 Typedef Documentation

9.35.3.1 GstTensorDebug

```
typedef struct _GstTensorDebug GstTensorDebug
```

Definition at line 41 of file gsttensor_debug.h.

9.35.3.2 GstTensorDebugClass

```
typedef struct _GstTensorDebugClass GstTensorDebugClass
```

Definition at line 42 of file gsttensor_debug.h.

9.35.4 Enumeration Type Documentation

9.35.4.1 tdbg_cap_mode

```
enum tdbg_cap_mode
```

Property "CAP" specification.

Enumerator

TDBG_CAP_DISABLED	No output for tensor-capability info
TDBG_CAP_SHOW_UPDATE	Output tensor-capability if there is a change in gstcap
TDBG_CAP_SHOW_UPDATE_F	Output tensor-capability if there is a change in dimensions even if gstcap has no changes. This happens with format=flexible or sparse
TDBG_CAP_SHOW_ALWAYS	

Definition at line 63 of file gsttensor_debug.h.

9.35.4.2 tdbg_meta_mode

`enum tdbg_meta_mode`

Property "META" specification.

Enumerator

TDBG_META_DISABLED	Don't bring up metadata of a stream
TDBG_META_TIMESTAMP	Enable timestamp info
TDBG_META_QUERYSERVER	Enable tensor-query-server related info

Definition at line 74 of file gsttensor_debug.h.

9.35.4.3 tdbg_output_mode

`enum tdbg_output_mode`

Property "OUTPUT" specification.

Enumerator

TDBG_OUTPUT_DISABLED	disable output.
TDBG_OUTPUT_CONSOLE_I	console/debug info
TDBG_OUTPUT_CONSOLE_W	console/debug warn
TDBG_OUTPUT_CONSOLE_E	console/debug error (non fatal)
TDBG_OUTPUT_GSTDBG_I	gst-debug info
TDBG_OUTPUT_GSTDBG_W	gst-debug warn
TDBG_OUTPUT_GSTDBG_E	gst-debug error (non fatal)
TDBG_OUTPUT_CIRCULARBUF	in circular buffer for later retrievals. (**Todo** NYI NOT_SUPPORTED)
TDBG_OUTPUT_FILEWRITE	Write to a file. (**Todo** NYI NOT_SUPPORTED)

Definition at line 47 of file gsttensor_debug.h.

9.35.5 Function Documentation

9.35.5.1 gst_tensor_debug_get_type()

```
GType gst_tensor_debug_get_type (
            void  )
```

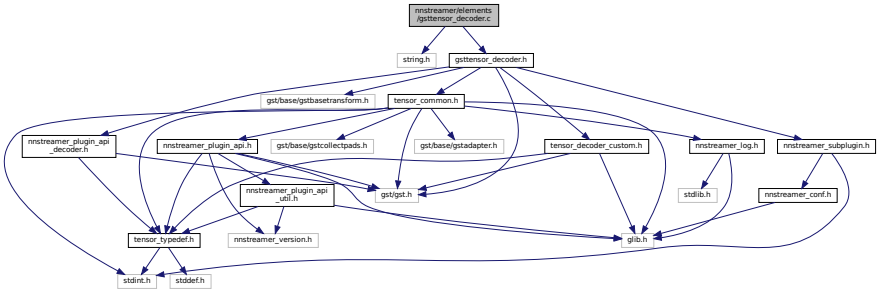
Get Type function required for gst elements.

9.36 nnstreamer/elements/gsttensor_decoder.c File Reference

GStreamer plugin to convert tensors (as a filter for other general neural network filters) to other media types.

```
#include <string.h>
#include "gsttensor_decoder.h"
```

Include dependency graph for gsttensor_decoder.c:



Macros

- #define DBG (!self->silent)

  Macro for debug mode.
- #define GST_CAT_DEFAULT gst_tensordec_debug
- #define DEFAULT_SILENT TRUE

  Flag to print minimized log.
- #define CAPS_STRING GST_TENSOR_CAP_DEFAULT ";" GST_TENSORS_CAP_DEFAULT ";" GST_TENSORS_FLEX_CAP_DEFAULT

  Support multi-tensor along with single-tensor as the input.
- #define gst_tensordec_parent_class parent_class
- #define gst_tensor_decoder_clean_plugin(self)

  Macro to clean sub-plugin data.
- #define PROP_MODE_OPTION(opnum) case PROP_MODE_OPTION

  A macro to process incoming per-mode option.
- #define PROP_READ_OPTION(opnum) case PROP_MODE_OPTION

  A macro to read per-mode option.

Enumerations

- enum {
  PROP_0, PROP_SILENT, PROP_MODE, PROP_MODE_OPTION1,
  PROP_MODE_OPTION2, PROP_MODE_OPTION3, PROP_MODE_OPTION4, PROP_MODE_OPTION5,
  PROP_MODE_OPTION6, PROP_MODE_OPTION7, PROP_MODE_OPTION8, PROP_MODE_OPTION9,
  PROP_SUBPLUGINS, PROP_CONFIG }

  Properties.

Functions

- GST_DEBUG_CATEGORY_STATIC (gst_tensordec_debug)
- G_DEFINE_TYPE (GstTensorDecoder, gst_tensordec, GST_TYPE_BASE_TRANSFORM)
- static void gst_tensordec_set_property (GObject *object, guint prop_id, const GValue *value, GParamSpec *pspec)

  Set property (GObject vmethod)
- static void gst_tensordec_get_property (GObject *object, guint prop_id, GValue *value, GParamSpec *pspec)

  Get property (GObject vmethod)
- static void gst_tensordec_class_finalize (GObject *object)

  Finalize instance (GObject vmethod)
- static GstFlowReturn gst_tensordec_transform (GstBaseTransform *trans, GstBuffer *inbuf, GstBuffer *outbuf)

  non-ip transform. required vmethod for BaseTransform class.
- static GstCaps * gst_tensordec_transform_caps (GstBaseTransform *trans, GstPadDirection direction, GstCaps *caps, GstCaps *filter)

  configure tensor-srcpad cap from "proposed" cap.
- static GstCaps * gst_tensordec_fixate_caps (GstBaseTransform *trans, GstPadDirection direction, GstCaps *caps, GstCaps *othercaps)

  fixate caps. required vmethod of BaseTransform
- static gboolean gst_tensordec_set_caps (GstBaseTransform *trans, GstCaps *incaps, GstCaps *outcaps)

  set caps. required vmethod of BaseTransform
- static gboolean gst_tensordec_transform_size (GstBaseTransform *trans, GstPadDirection direction, GstCaps *caps, gsize size, GstCaps *othercaps, gsize *othersize)

  Tell the framework the required size of buffer based on the info of the other side pad. optional vmethod of BaseTransform.
- static gboolean nnstreamer_decoder_validate (const GstTensorDecoderDef *decoder)

  Validate decoder sub-plugin's data.
- int nnstreamer_decoder_probe (GstTensorDecoderDef *decoder)

  Decoder's sub-plugin should call this function to register itself.
- void nnstreamer_decoder_exit (const char *name)

  Decoder's sub-plugin may call this to unregister itself.
- const GstTensorDecoderDef * nnstreamer_decoder_find (const char *name)

  Find decoder sub-plugin with the name.
- void nnstreamer_decoder_set_custom_property_desc (const char *name, const char *prop,...)

  set custom property description for tensor decoder sub-plugin
- static GstCaps * gst_tensordec_media_caps_from_tensor (GstTensorDecoder *self, const GstTensorsConfig *config)

  Get media caps from tensor config.
- static GstCaps * gst_tensordec_media_caps_from_structure (GstTensorDecoder *self, const GstStructure *structure)

  Parse structure and return media caps.

- static gboolean gst_tensordec_check_consistency (GstTensorDecoder *self, GstTensorsConfig *config)

  Check tensor config is consistent.
- static void gst_tensordec_class_init (GstTensorDecoderClass *klass)

  initialize the tensordec's class
- static void gst_tensordec_init (GstTensorDecoder *self)

  initialize the new element instantiate pads and add them to element set pad callback functions initialize instance structure
- static gboolean gst_tensordec_process_plugin_options (GstTensorDecoder *self, guint opnum)

  Process plugin (self->decoder) with given options if available.
- g_free (self->option[(opnum) - 1])

  *opnum: *
- if (!gst_tensordec_process_plugin_options(self,(opnum) - 1)) GST_ERROR_OBJECT(self
- Configuring option for tensor decoder failed (option %d=%s)"
- g_value_set_string (value, self->option[opnum - 1])

  *opnum: *
- static gboolean gst_tensordec_configure (GstTensorDecoder *self, const GstCaps *in_caps, const GstCaps *out_caps)

  Configure tensor metadata from sink caps.
- int nnstreamer_decoder_custom_register (const gchar *name, tensor_decoder_custom func, void *data)

  Registers a callback for tensor_decoder custom condition.
- int nnstreamer_decoder_custom_unregister (const gchar *name)

  Unregisters a callback for tensor_decoder custom condition.

Variables

- static GstStaticPadTemplate sink_factory

  The capabilities of the inputs.
- static GstStaticPadTemplate src_factory

  The capabilities of the outputs.
- self option [(opnum) - 1] = g_value_dup_string (value)
- Configuring option for tensor decoder opnum

9.36.1 Detailed Description

GStreamer plugin to convert tensors (as a filter for other general neural network filters) to other media types.

GStreamer / NNStreamer tensor_decoder main Copyright (C) 2005 Thomas Vander Stichele thomas@apestaart.↩org Copyright (C) 2005 Ronald S. Bultje rbultje@ronald.bitfreak.net Copyright (C) 2018 Jijoong Moon jijoong.moon@samsung.com

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Library General Public License as published by the Free Software Foundation; version 2.1 of the License.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Library General Public License for more details.

Date

26 Mar 2018

See also

https://github.com/nnstreamer/nnstreamer

Author

Jijoong Moon jijoong.moon@samsung.com

Bug gst_tensordec_transform_size () may be incorrect if direction is SINK.

If configured = TRUE, it holds TRUE until exit. What if configuration changes in run-time?

9.36.2 Macro Definition Documentation

9.36.2.1 CAPS_STRING

```
#define CAPS_STRING GST_TENSOR_CAP_DEFAULT ";" GST_TENSORS_CAP_DEFAULT ";" GST_TENSORS_FLEX_CAP_DEFAULT
```

Support multi-tensor along with single-tensor as the input.

Definition at line 89 of file gsttensor_decoder.c.

9.36.2.2 DBG

```
#define DBG (!self->silent)
```

Macro for debug mode.

SECTION:element-tensordec

A filter that converts tensor stream for NN frameworks to media stream. The input is always in the format of other/tensor

<refsect2> <title>Example launch line</title> |[gst-launch -v -m fakesink ! tensor_decoder ! fakesrc silent=T↩ RUE]| </refsect2>

Definition at line 54 of file gsttensor_decoder.c.

9.36.2.3 DEFAULT_SILENT

```
#define DEFAULT_SILENT TRUE
```

Flag to print minimized log.

Definition at line 84 of file gsttensor_decoder.c.

9.36.2.4 GST_CAT_DEFAULT

```
#define GST_CAT_DEFAULT gst_tensordec_debug
```

Definition at line 58 of file gsttensor_decoder.c.

9.36.2.5 gst_tensor_decoder_clean_plugin

```
#define gst_tensor_decoder_clean_plugin(
              self )
```

Value:

```
    do { \
    if (self->decoder) { \
      if (self->decoder->exit) \
        self->decoder->exit (&self->plugin_data); \
      else \
        g_free (self->plugin_data); \
      self->plugin_data = NULL; \
    } \
  } while (0)
```

Macro to clean sub-plugin data.

Definition at line 200 of file gsttensor_decoder.c.

9.36.2.6 gst_tensordec_parent_class

```
#define gst_tensordec_parent_class parent_class
```

Definition at line 107 of file gsttensor_decoder.c.

9.36.2.7 PROP_MODE_OPTION

```
#define PROP_MODE_OPTION(
              opnum ) case PROP_MODE_OPTION
```

A macro to process incoming per-mode option.

Parameters

in	*opnum*	The option number (1 to TensorDecMaxOpNum)

Definition at line 457 of file gsttensor_decoder.c.

9.36.2.8 PROP_READ_OPTION

```
#define PROP_READ_OPTION(
              opnum ) case PROP_MODE_OPTION
```

A macro to read per-mode option.

Parameters

in	*opnum*	The option number (1 to TensorDecMaxOpNum)

Definition at line 556 of file gsttensor_decoder.c.

9.36.3 Enumeration Type Documentation

9.36.3.1 anonymous enum

```
anonymous enum
```

Properties.

Enumerator

PROP_0	
PROP_SILENT	
PROP_MODE	
PROP_MODE_OPTION1	
PROP_MODE_OPTION2	
PROP_MODE_OPTION3	
PROP_MODE_OPTION4	
PROP_MODE_OPTION5	
PROP_MODE_OPTION6	
PROP_MODE_OPTION7	
PROP_MODE_OPTION8	
PROP_MODE_OPTION9	
PROP_SUBPLUGINS	
PROP_CONFIG	

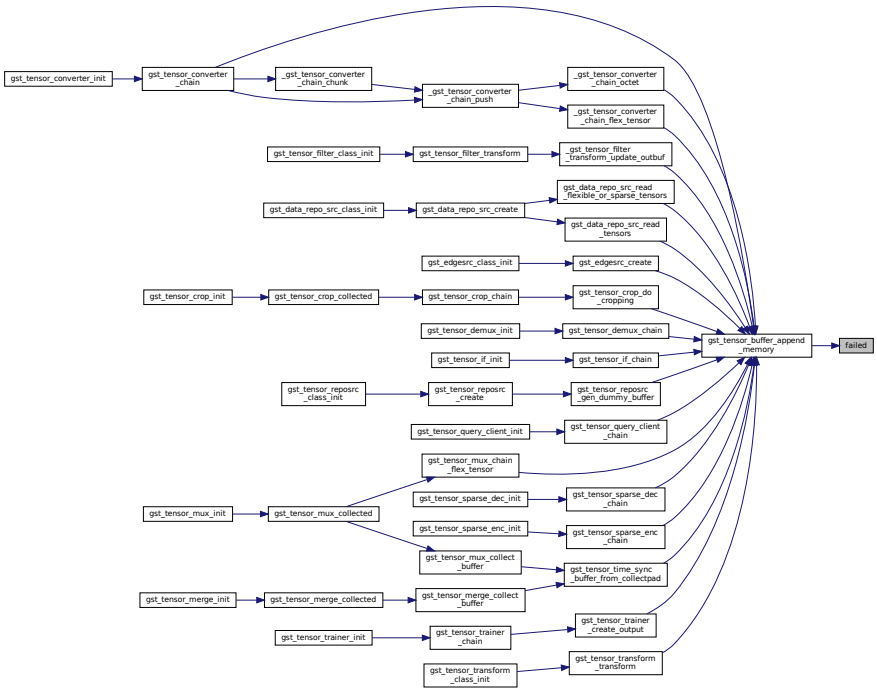
Definition at line 63 of file gsttensor_decoder.c.

9.36.4 Function Documentation

9.36.4.1 failed()

```
Configuring option for tensor decoder failed (
            option % d = %s )
```

Here is the caller graph for this function:



9.36.4.2 G_DEFINE_TYPE()

```
G_DEFINE_TYPE (
            GstTensorDecoder ,
            gst_tensordec ,
            GST_TYPE_BASE_TRANSFORM  )
```

9.36.4.3 g_free()

```
g_free (
            self-> option[(opnum) - 1] )
```

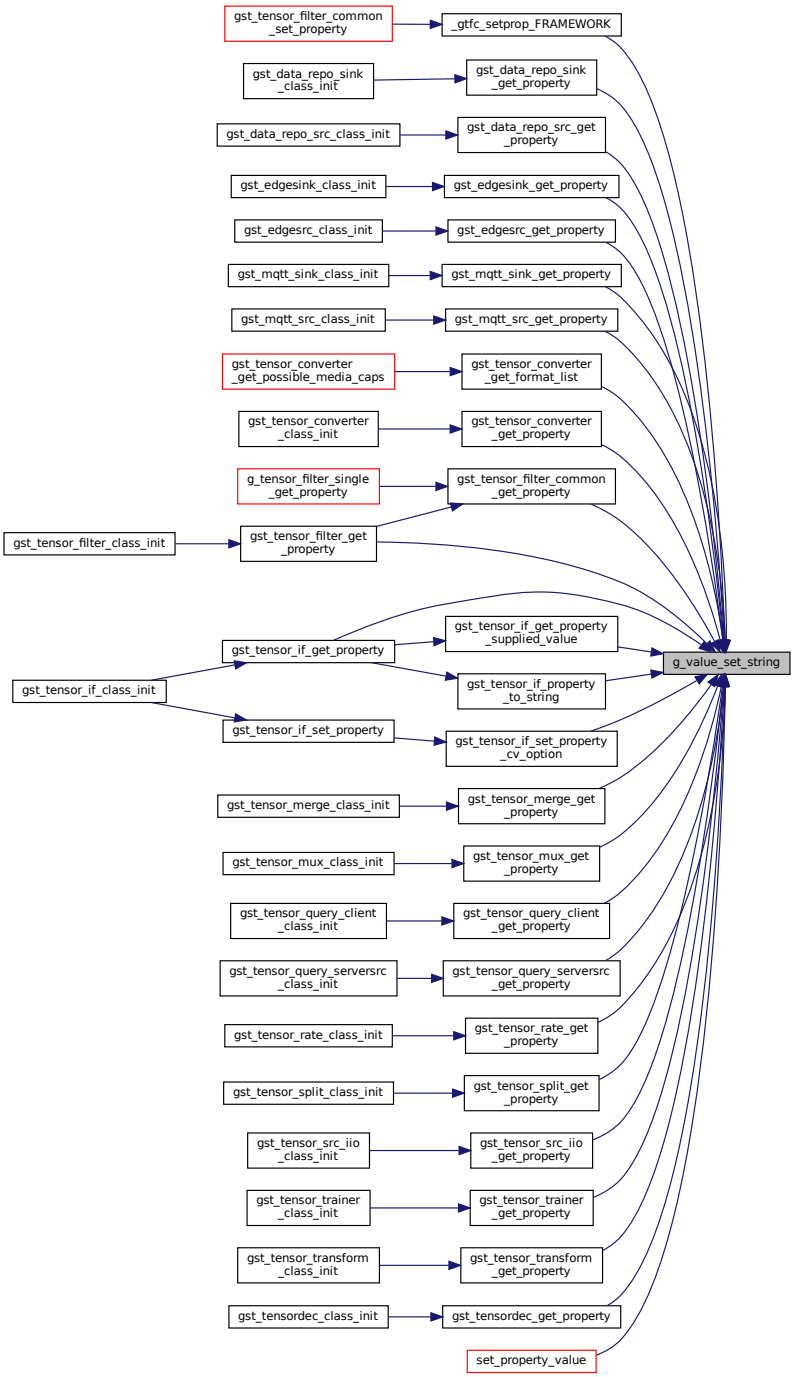
opnum: \

9.36.4.4 g_value_set_string()

```
g_value_set_string (
            value ,
            self-> option[opnum - 1] )
```

opnum: \

Here is the caller graph for this function:



9.36.4.5 GST_DEBUG_CATEGORY_STATIC()

```
GST_DEBUG_CATEGORY_STATIC (
            gst_tensordec_debug  )
```

9.36.4.6 gst_tensordec_check_consistency()

```
static gboolean gst_tensordec_check_consistency (
            GstTensorDecoder * self,
            GstTensorsConfig * config )  [static]
```

Check tensor config is consistent.

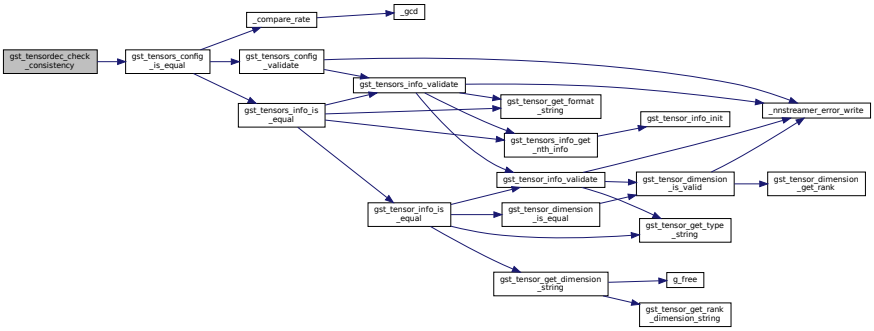
Parameters

self	"this" pointer to check consistency
t_info	newly configured tensor metadata

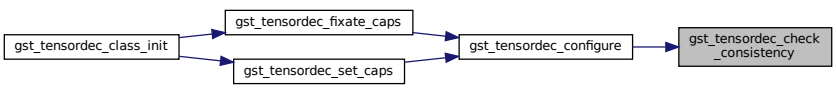
not configured yet

Definition at line 269 of file gsttensor_decoder.c.

Here is the call graph for this function:



Here is the caller graph for this function:



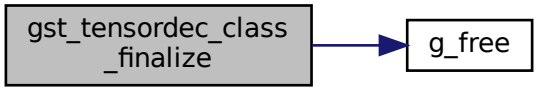
9.36.4.7 gst_tensordec_class_finalize()

```
static void gst_tensordec_class_finalize (
            GObject * object )  [static]
```

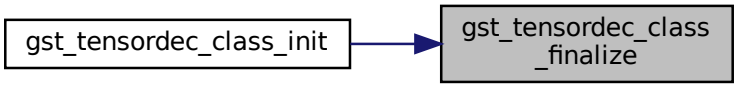
Finalize instance (GObject vmethod)

Definition at line 618 of file gsttensor_decoder.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.36.4.8 gst_tensordec_class_init()

```
static void gst_tensordec_class_init (
            GstTensorDecoderClass * klass )  [static]
```

initialize the tensordec's class

Refer: https://gstreamer.freedesktop.org/documentation/design/element-transform.↩
html

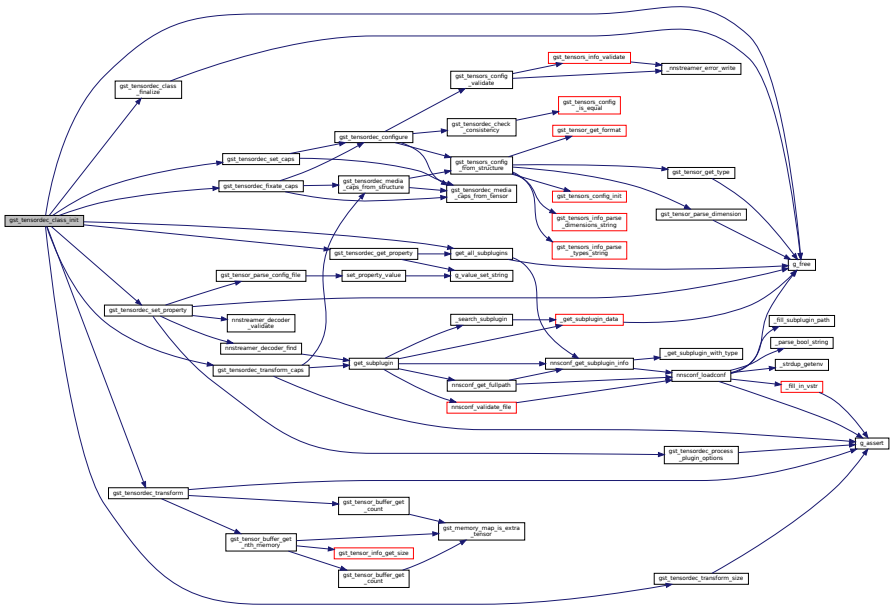
Processing units

Negotiation units

Allocation units

Definition at line 287 of file gsttensor_decoder.c.

Here is the call graph for this function:



9.36.4.9 gst_tensordec_configure()

```
static gboolean gst_tensordec_configure (
            GstTensorDecoder * self,
            const GstCaps * in_caps,
            const GstCaps * out_caps )  [static]
```

Configure tensor metadata from sink caps.

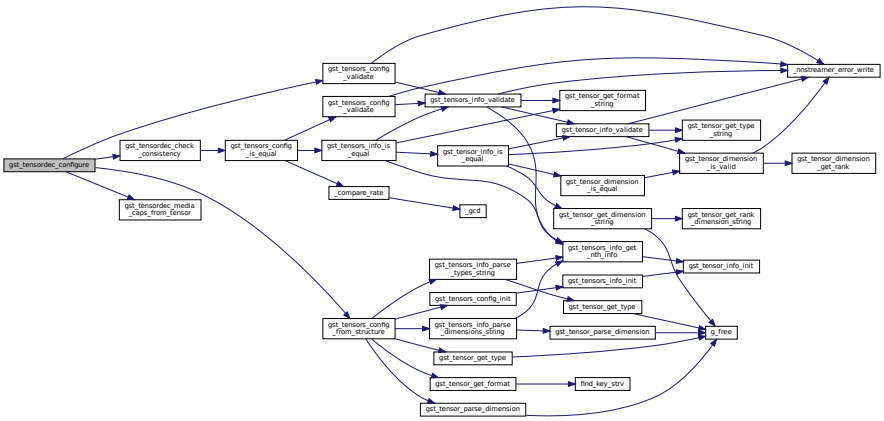
This caps is coming from tensor

If previous input configuration is set and is not compatible with incoming caps, get possible media caps from sub-plugin and change input configuration.

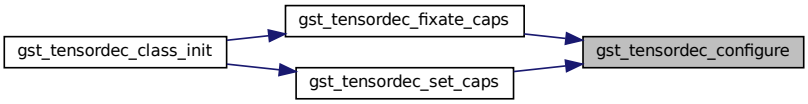
Check if outcaps is compatible with new caps

Definition at line 641 of file gsttensor_decoder.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.36.4.10 gst_tensordec_fixate_caps()

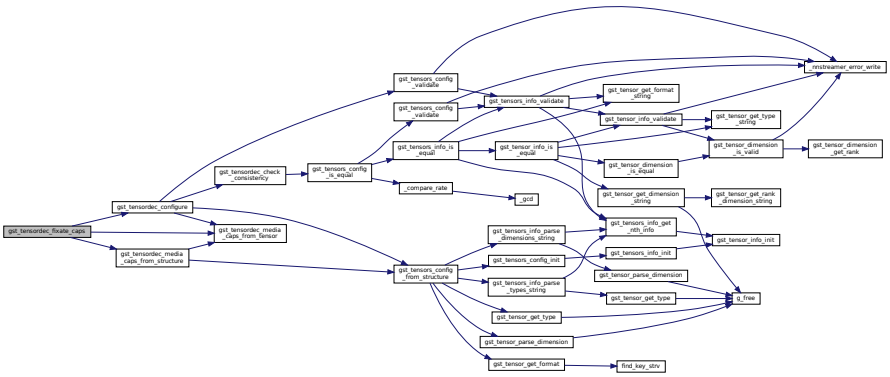
```
static GstCaps * gst_tensordec_fixate_caps (
            GstBaseTransform * trans,
            GstPadDirection direction,
            GstCaps * caps,
            GstCaps * othercaps )  [static]
```

fixate caps. required vmethod of BaseTransform

In gst_tensordec_transform_caps, we have refused to specify caps if the direction is GST_PAD_SRC. Thus, gstreamer shouldn't fixate it with GST_PAD_SRC. If this happens, it's either internal logic error or GST bUG.

Definition at line 852 of file gsttensor_decoder.c.

Here is the call graph for this function:



Here is the caller graph for this function:

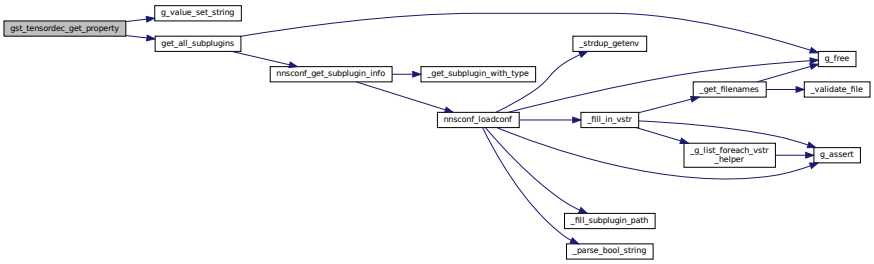
9.36.4.11 gst_tensordec_get_property()

```
static break void gst_tensordec_get_property (
            GObject * object,
            guint prop_id,
            GValue * value,
            GParamSpec * pspec )  [static]
```

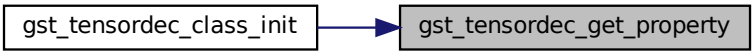
Get property (GObject vmethod)

Definition at line 565 of file gsttensor_decoder.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.36.4.12 gst_tensordec_init()

```
static void gst_tensordec_init (
            GstTensorDecoder * self )  [static]
```

initialize the new element instantiate pads and add them to element set pad callback functions initialize instance structure

Definition at line 416 of file gsttensor_decoder.c.

Here is the call graph for this function:

9.36.4.13 gst_tensordec_media_caps_from_structure()

```
static GstCaps* gst_tensordec_media_caps_from_structure (
            GstTensorDecoder * self,
            const GstStructure * structure )  [static]
```

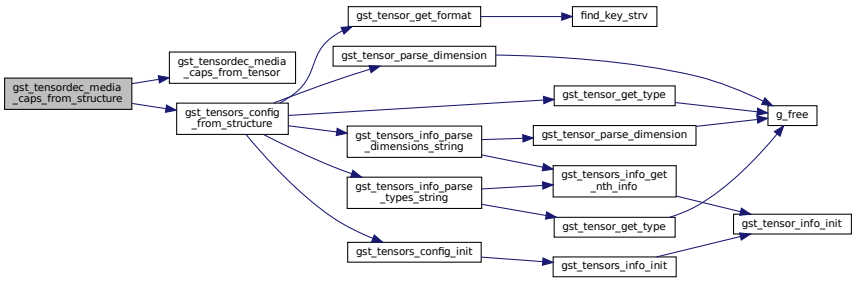
Parse structure and return media caps.

Parameters

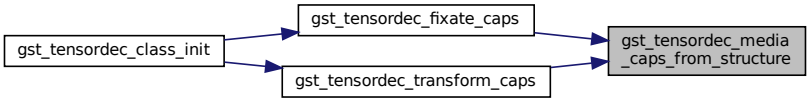
self	"this" pointer
structure	structure to be interpreted

Definition at line 245 of file gsttensor_decoder.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.36.4.14 gst_tensordec_media_caps_from_tensor()

```
static GstCaps* gst_tensordec_media_caps_from_tensor (
            GstTensorDecoder * self,
            const GstTensorsConfig * config )  [static]
```

Get media caps from tensor config.

Parameters

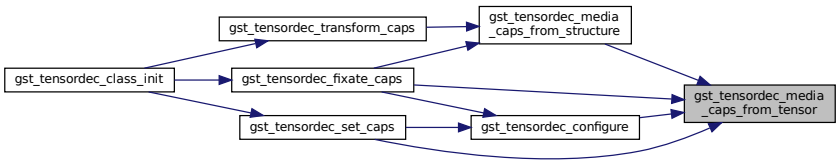
self	"this" pointer
config	tensor config info

Returns

caps for media type

Definition at line 217 of file gsttensor_decoder.c.

Here is the caller graph for this function:



9.36.4.15 gst_tensordec_process_plugin_options()

```
static gboolean gst_tensordec_process_plugin_options (
            GstTensorDecoder * self,
            guint opnum )  [static]
```

Process plugin (self->decoder) with given options if available.

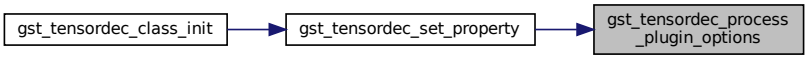
Return values

FALSE	if error. TRUE if OK (or SKIP)

Definition at line 440 of file gsttensor_decoder.c.

Here is the call graph for this function:

Here is the caller graph for this function:



9.36.4.16 gst_tensordec_set_caps()

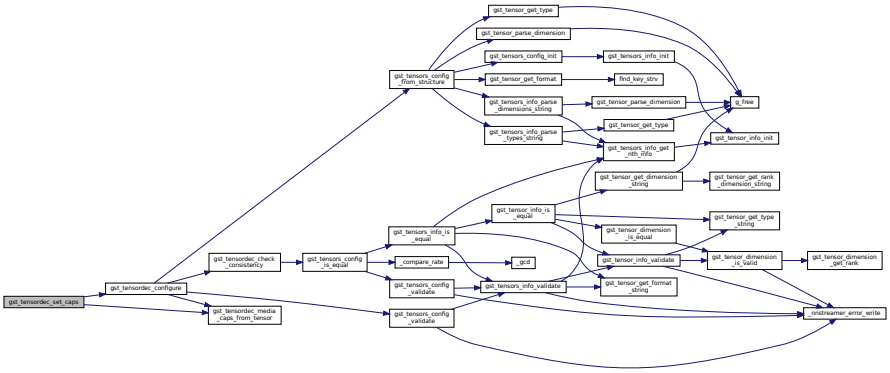
```
static gboolean gst_tensordec_set_caps (
            GstBaseTransform * trans,
            GstCaps * incaps,
            GstCaps * outcaps )  [static]
```

set caps. required vmethod of BaseTransform

Check if outcaps ==equivalent== supposed

Definition at line 913 of file gsttensor_decoder.c.

Here is the call graph for this function:



Here is the caller graph for this function:

9.36.4.17 gst_tensordec_set_property()

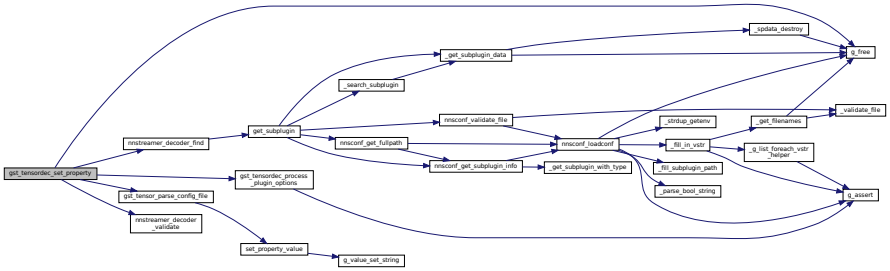
```
static break void gst_tensordec_set_property (
            GObject * object,
            guint prop_id,
            const GValue * value,
            GParamSpec * pspec )  [static]
```

Set property (GObject vmethod)

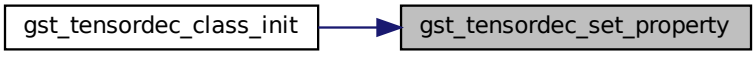
GObject vmethod implementations

Definition at line 470 of file gsttensor_decoder.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.36.4.18 gst_tensordec_transform()

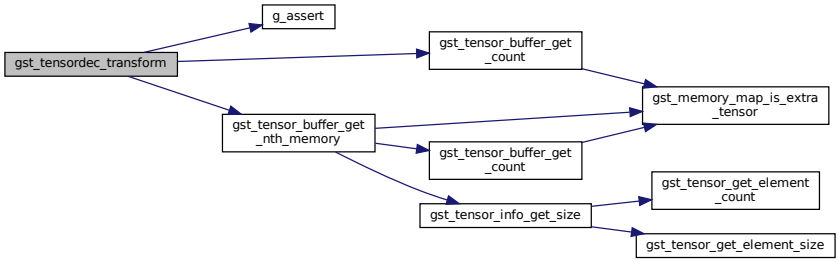
```
static GstFlowReturn gst_tensordec_transform (
            GstBaseTransform * trans,
            GstBuffer * inbuf,
            GstBuffer * outbuf )  [static]
```

non-ip transform. required vmethod for BaseTransform class.

GstBaseTransform vmethod implementations Internal logic error. Negotiation process should prevent this!

Definition at line 696 of file gsttensor_decoder.c.

Here is the call graph for this function:



Here is the caller graph for this function:

9.36.4.19 gst_tensordec_transform_caps()

```
static GstCaps * gst_tensordec_transform_caps (
            GstBaseTransform * trans,
            GstPadDirection direction,
            GstCaps * caps,
            GstCaps * filter )  [static]
```

configure tensor-srcpad cap from "proposed" cap.

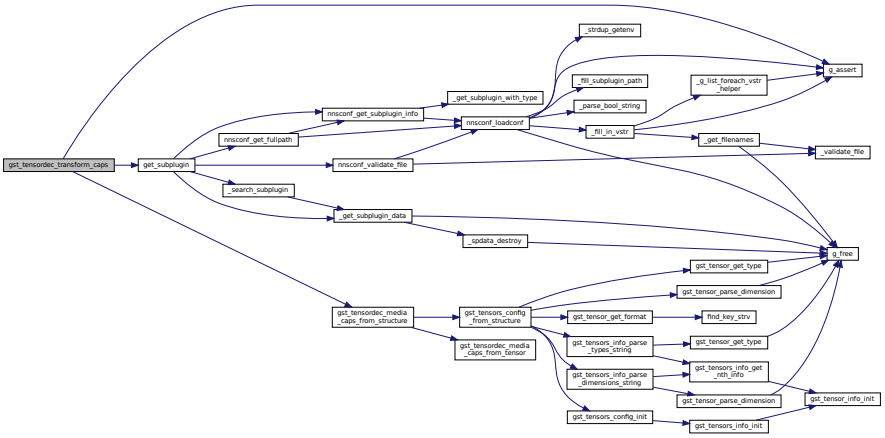
@trans ("this" pointer) @direction (why do we need this?) @caps sinkpad cap @filter this element's cap (don't know specifically.) caps = sinkpad (other/tensor) return = srcpad (media)

caps = srcpad (media) return = sinkpad (other/tensor)

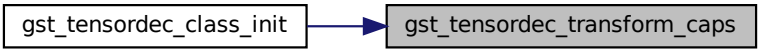
Todo We may do more specific actions here

Definition at line 786 of file gsttensor_decoder.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.36.4.20 gst_tensordec_transform_size()

```
static gboolean gst_tensordec_transform_size (
            GstBaseTransform * trans,
            GstPadDirection direction,
            GstCaps * caps,
            gsize size,
            GstCaps * othercaps,
            gsize * othersize )  [static]
```

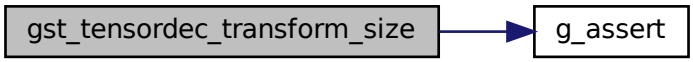
Tell the framework the required size of buffer based on the info of the other side pad. optional vmethod of Base↩ Transform.

This is called when non-ip mode is used.

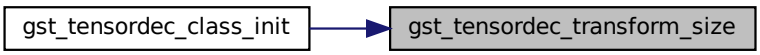
Todo If direction = SRC, you may need different interpretation!

Definition at line 945 of file gsttensor_decoder.c.

Here is the call graph for this function:



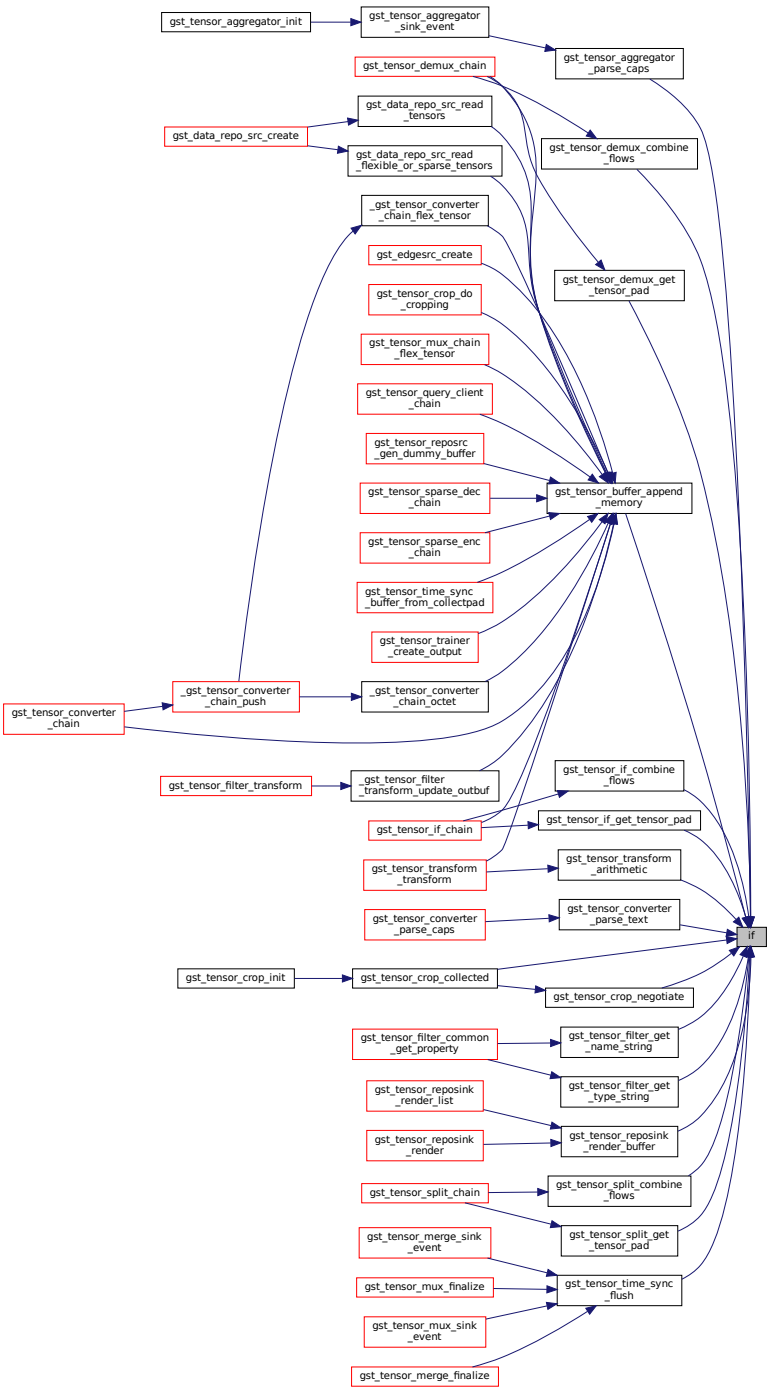
Here is the caller graph for this function:



9.36.4.21 if()

```
if (
             !  gst_tensordec_process_plugin_optionsself,(opnum) - 1 )
```

Here is the caller graph for this function:



9.36.4.22 nnstreamer_decoder_custom_register()

```
int nnstreamer_decoder_custom_register (
            const gchar * name,
```

```
tensor_decoder_custom func,
void * data )
```

Registers a callback for tensor_decoder custom condition.

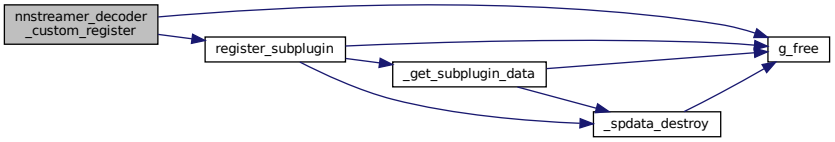
Register the custom callback function.

Returns

0 if success. -ERRNO if error.

Definition at line 972 of file gsttensor_decoder.c.

Here is the call graph for this function:



9.36.4.23 nnstreamer_decoder_custom_unregister()

```
int nnstreamer_decoder_custom_unregister (
            const gchar * name )
```

Unregisters a callback for tensor_decoder custom condition.

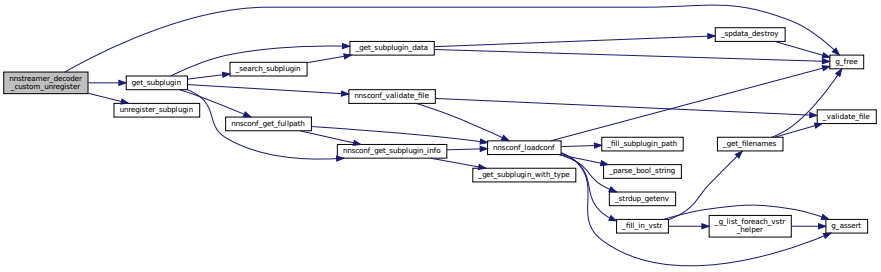
Unregister the custom callback function.

Returns

0 if success. -ERRNO if error.

Definition at line 998 of file gsttensor_decoder.c.

Here is the call graph for this function:



9.36.4.24 nnstreamer_decoder_exit()

```
void nnstreamer_decoder_exit (
            const char * name )
```

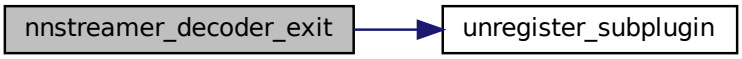
Decoder's sub-plugin may call this to unregister itself.

Parameters

in	*name*	The name of decoder sub-plugin.

Definition at line 166 of file gsttensor_decoder.c.

Here is the call graph for this function:



9.36.4.25 nnstreamer_decoder_find()

```
const GstTensorDecoderDef* nnstreamer_decoder_find (
            const char * name )
```

Find decoder sub-plugin with the name.

Parameters

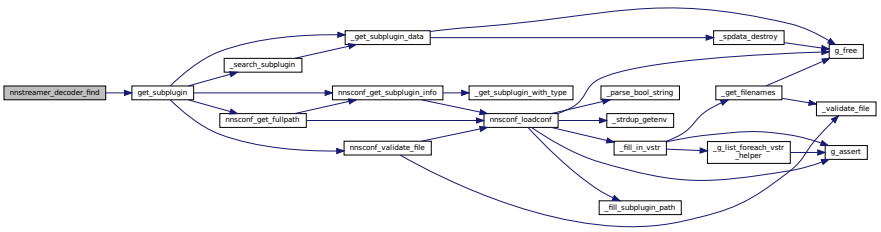
in	*name*	The name of decoder sub-plugin.

Returns

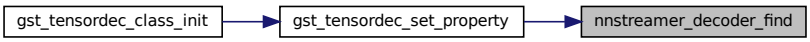
NULL if not found or the sub-plugin object has an error.

Definition at line 177 of file gsttensor_decoder.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.36.4.26 nnstreamer_decoder_probe()

```
int nnstreamer_decoder_probe (
            GstTensorDecoderDef * decoder )
```

Decoder's sub-plugin should call this function to register itself.

Parameters

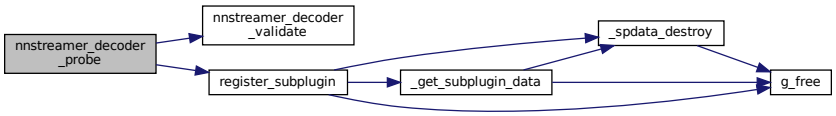
in	*decoder*	Decoder sub-plugin to be registered.

Returns

TRUE if registered. FALSE is failed or duplicated.

Definition at line 155 of file gsttensor_decoder.c.

Here is the call graph for this function:



9.36.4.27 nnstreamer_decoder_set_custom_property_desc()

```
void nnstreamer_decoder_set_custom_property_desc (
            const char * name,
            const char * prop,
             ...  )
```

set custom property description for tensor decoder sub-plugin

Definition at line 186 of file gsttensor_decoder.c.

Here is the call graph for this function:

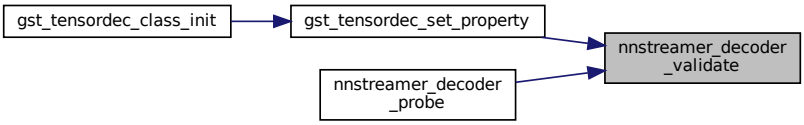
9.36.4.28 nnstreamer_decoder_validate()

```
static gboolean nnstreamer_decoder_validate (
            const GstTensorDecoderDef * decoder )  [static]
```

Validate decoder sub-plugin's data.

Definition at line 134 of file gsttensor_decoder.c.

Here is the caller graph for this function:



9.36.5 Variable Documentation

9.36.5.1 opnum

```
Configuring option for tensor decoder opnum
```

Definition at line 463 of file gsttensor_decoder.c.

9.36.5.2 option

```
Configuring option for tensor decoder self option = g_value_dup_string (value)
```

Definition at line 460 of file gsttensor_decoder.c.

9.36.5.3 sink_factory

```
GstStaticPadTemplate sink_factory  [static]
```

Initial value:
```
= GST_STATIC_PAD_TEMPLATE ("sink",
    GST_PAD_SINK,
    GST_PAD_ALWAYS,
    GST_STATIC_CAPS (CAPS_STRING))
```

The capabilities of the inputs.

Definition at line 94 of file gsttensor_decoder.c.

9.36.5.4 src_factory

```
GstStaticPadTemplate src_factory  [static]
```

Initial value:
```
= GST_STATIC_PAD_TEMPLATE ("src",
    GST_PAD_SRC,
    GST_PAD_ALWAYS,
    GST_STATIC_CAPS ("ANY"))
```

The capabilities of the outputs.

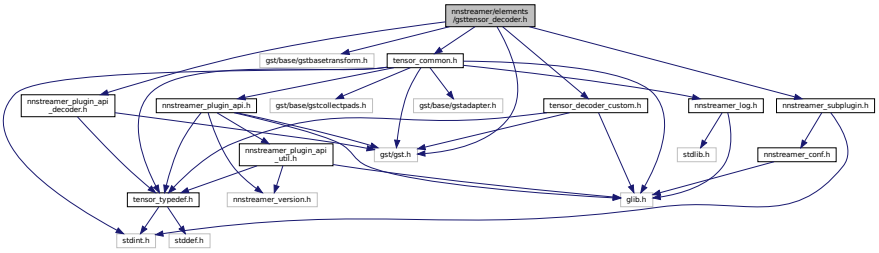
Definition at line 102 of file gsttensor_decoder.c.

9.37 nnstreamer/elements/gsttensor_decoder.h File Reference

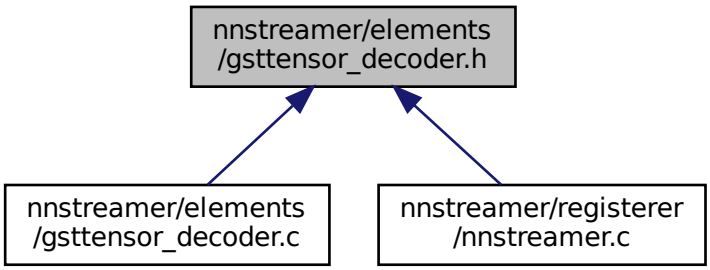
GStreamer plugin to convert tensors to media types.

```
#include <gst/gst.h>
#include <gst/base/gstbasetransform.h>
#include "tensor_common.h"
#include "nnstreamer_subplugin.h"
#include "nnstreamer_plugin_api_decoder.h"
#include "tensor_decoder_custom.h"
```

Include dependency graph for gsttensor_decoder.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct decoder_custom_cb_s
- struct _GstTensorDecoder

  Internal data structure for tensordec instances.
- struct _GstTensorDecoderClass

  GstTensorDecoderClass inherits GstBaseTransformClass.

Macros

- #define GST_TYPE_TENSOR_DECODER (gst_tensordec_get_type())
- #define GST_TENSOR_DECODER(obj) (G_TYPE_CHECK_INSTANCE_CAST((obj),GST_TYPE_TENSOR_DECODER,GstT
- #define GST_TENSOR_DECODER_CLASS(klass) (G_TYPE_CHECK_CLASS_CAST((klass),GST_TYPE_TENSOR_DECOD
- #define GST_IS_TENSOR_DECODER(obj) (G_TYPE_CHECK_INSTANCE_TYPE((obj),GST_TYPE_TENSOR_DECODER))
- #define GST_IS_TENSOR_DECODER_CLASS(klass) (G_TYPE_CHECK_CLASS_TYPE((klass),GST_TYPE_TENSOR_DEC
- #define GST_TENSOR_DECODER_CAST(obj) ((GstTensorDecoder *)(obj))
- #define TensorDecMaxOpNum (9)

Typedefs

- typedef struct _GstTensorDecoder GstTensorDecoder
- typedef struct _GstTensorDecoderClass GstTensorDecoderClass

Functions

- GType gst_tensordec_get_type (void)

  Get Type function required for gst elements.

9.37.1 Detailed Description

GStreamer plugin to convert tensors to media types.

GStreamer / NNStreamer tensor_decoder header Copyright (C) 2005 Thomas Vander Stichele thomas@apestaart.org Copyright (C) 2005 Ronald S. Bultje rbultje@ronald.bitfreak.net Copyright (C) 2018 Jijoong Moon jijoong.moon@samsung.com

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Library General Public License as published by the Free Software Foundation; version 2.1 of the License.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Library General Public License for more details.

Date

26 Mar 2018

See also

https://github.com/nnstreamer/nnstreamer

Author

Jijoong Moon jijoong.moon@samsung.com

Bug No known bugs except for NYI items

9.37.2 Macro Definition Documentation

9.37.2.1 GST_IS_TENSOR_DECODER

```
#define GST_IS_TENSOR_DECODER(
              obj ) (G_TYPE_CHECK_INSTANCE_TYPE((obj),GST_TYPE_TENSOR_DECODER))
```

Definition at line 48 of file gsttensor_decoder.h.

9.37.2.2 GST_IS_TENSOR_DECODER_CLASS

```
#define GST_IS_TENSOR_DECODER_CLASS(
              klass ) (G_TYPE_CHECK_CLASS_TYPE((klass),GST_TYPE_TENSOR_DECODER))
```

Definition at line 50 of file gsttensor_decoder.h.

9.37.2.3 GST_TENSOR_DECODER

```
#define GST_TENSOR_DECODER(
              obj ) (G_TYPE_CHECK_INSTANCE_CAST((obj),GST_TYPE_TENSOR_DECODER,GstTensorDecoder))
```

Definition at line 44 of file gsttensor_decoder.h.

9.37.2.4 GST_TENSOR_DECODER_CAST

```
#define GST_TENSOR_DECODER_CAST(
              obj ) ((GstTensorDecoder *)(obj))
```

Definition at line 52 of file gsttensor_decoder.h.

9.37.2.5 GST_TENSOR_DECODER_CLASS

```
#define GST_TENSOR_DECODER_CLASS(
              klass ) (G_TYPE_CHECK_CLASS_CAST((klass),GST_TYPE_TENSOR_DECODER,GstTensorDecoderClass))
```

Definition at line 46 of file gsttensor_decoder.h.

9.37.2.6 GST_TYPE_TENSOR_DECODER

```
#define GST_TYPE_TENSOR_DECODER (gst_tensordec_get_type())
```

Definition at line 42 of file gsttensor_decoder.h.

9.37.2.7 TensorDecMaxOpNum

```
#define TensorDecMaxOpNum (9)
```

Definition at line 62 of file gsttensor_decoder.h.

9.37.3 Typedef Documentation

9.37.3.1 GstTensorDecoder

```
typedef struct _GstTensorDecoder GstTensorDecoder
```

Definition at line 54 of file gsttensor_decoder.h.

9.37.3.2 GstTensorDecoderClass

```
typedef struct _GstTensorDecoderClass GstTensorDecoderClass
```

Definition at line 55 of file gsttensor_decoder.h.

9.37.4 Function Documentation

9.37.4.1 gst_tensordec_get_type()

```
GType gst_tensordec_get_type (
            void  )
```

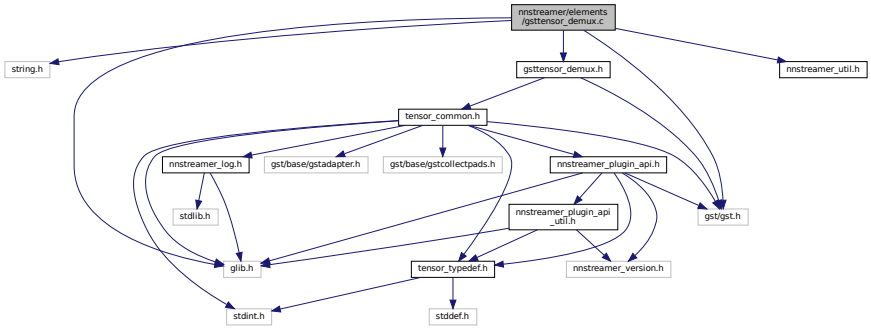
Get Type function required for gst elements.

9.38 nnstreamer/elements/gsttensor_demux.c File Reference

GStreamer plugin to demux tensors (as a filter for other general neural network filters)

```
#include <string.h>
#include <gst/gst.h>
#include <glib.h>
#include <nnstreamer_util.h>
#include "gsttensor_demux.h"
```

Include dependency graph for gsttensor_demux.c:



Macros

- #define GST_CAT_DEFAULT gst_tensor_demux_debug
- #define CAPS_STRING_SINK GST_TENSORS_CAP_MAKE ("{ static, flexible }")

  Default caps string for sink pad.
- #define CAPS_STRING_SRC GST_TENSOR_CAP_DEFAULT ";" GST_TENSORS_CAP_MAKE ("{ static, flexible }")

  Default caps string for src pad.
- #define gst_tensor_demux_parent_class parent_class

Enumerations

- enum { PROP_0, PROP_SILENT, PROP_TENSORPICK }

Functions

- GST_DEBUG_CATEGORY_STATIC (gst_tensor_demux_debug)
- static GstFlowReturn gst_tensor_demux_chain (GstPad *pad, GstObject *parent, GstBuffer *buf)

  chain function for sink (gst element vmethod)
- static gboolean gst_tensor_demux_event (GstPad *pad, GstObject *parent, GstEvent *event)

  event function for sink (gst element vmethod)
- static GstStateChangeReturn gst_tensor_demux_change_state (GstElement *element, GstStateChange transition)

  change state (gst element vmethod)
- static void gst_tensor_demux_set_property (GObject *object, guint prop_id, const GValue *value, GParam↩Spec *pspec)

  Get property (gst element vmethod)
- static void gst_tensor_demux_get_property (GObject *object, guint prop_id, GValue *value, GParamSpec *pspec)

  Get property (gst element vmethod)
- static void gst_tensor_demux_dispose (GObject *object)

  dispose function for tensor demux (gst element vmethod)
- G_DEFINE_TYPE (GstTensorDemux, gst_tensor_demux, GST_TYPE_ELEMENT)
- static void gst_tensor_demux_class_init (GstTensorDemuxClass *klass)

  initialize the tensor_demux's class
- static void gst_tensor_demux_init (GstTensorDemux *tensor_demux)

  initialize the new element instantiate pads and add them to element set pad callback functions initialize instance structure
- static void gst_tensor_demux_remove_src_pads (GstTensorDemux *tensor_demux)

  function to remove srcpad list
- static gboolean gst_tensor_demux_parse_caps (GstTensorDemux *tensor_demux, GstCaps *caps)

  Parse caps and configure tensors info.
- static gboolean gst_tensor_demux_get_tensor_config (GstTensorDemux *tensor_demux, GstTensorsConfig *config, const guint nth, const guint total)

  Get tensor config info from configured tensors.
- static GstTensorPad * gst_tensor_demux_get_tensor_pad (GstTensorDemux *tensor_demux, gboolean *created, const guint nth, const guint total)

  Checking if the source pad is created and if not, create TensorPad.
- static GstFlowReturn gst_tensor_demux_combine_flows (GstTensorDemux *tensor_demux, GstTensorPad *pad, GstFlowReturn ret)

  Check the status among sources in demux.

Variables

- static GstStaticPadTemplate src_templ

  the capabilities of the inputs and outputs. describe the real formats here.
- static GstStaticPadTemplate sink_templ

9.38.1 Detailed Description

GStreamer plugin to demux tensors (as a filter for other general neural network filters)

GStreamer Copyright (C) 2005 Thomas Vander Stichele thomas@apestaart.org Copyright (C) 2005 Ronald S. Bultje rbultje@ronald.bitfreak.net Copyright (C) 2018 Jijoong Moon jijoong.moon@samsung.com

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Library General Public License as published by the Free Software Foundation; version 2.1 of the License.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Library General Public License for more details.

Date

03 July 2018

See also

https://github.com/nnstreamer/nnstreamer

Author

Jijoong Moon jijoong.moon@samsung.com

Bug No known bugs except for NYI items

9.38.2 Macro Definition Documentation

9.38.2.1 CAPS_STRING_SINK

```
#define CAPS_STRING_SINK GST_TENSORS_CAP_MAKE ("{ static, flexible }")
```

Default caps string for sink pad.

Definition at line 78 of file gsttensor_demux.c.

9.38.2.2 CAPS_STRING_SRC

```
#define CAPS_STRING_SRC GST_TENSOR_CAP_DEFAULT ";" GST_TENSORS_CAP_MAKE ("{ static, flexible
}")
```

Default caps string for src pad.

Definition at line 83 of file gsttensor_demux.c.

9.38.2.3 GST_CAT_DEFAULT

```
#define GST_CAT_DEFAULT gst_tensor_demux_debug
```

Definition at line 73 of file gsttensor_demux.c.

9.38.2.4 gst_tensor_demux_parent_class

```
#define gst_tensor_demux_parent_class parent_class
```

Definition at line 119 of file gsttensor_demux.c.

9.38.3 Enumeration Type Documentation

9.38.3.1 anonymous enum

```
anonymous enum
```

Enumerator

PROP_0	
PROP_SILENT	
PROP_TENSORPICK	

Definition at line 85 of file gsttensor_demux.c.

9.38.4 Function Documentation

9.38.4.1 G_DEFINE_TYPE()

```
G_DEFINE_TYPE (
            GstTensorDemux ,
            gst_tensor_demux ,
            GST_TYPE_ELEMENT  )
```

9.38.4.2 GST_DEBUG_CATEGORY_STATIC()

```
GST_DEBUG_CATEGORY_STATIC (
            gst_tensor_demux_debug  )
```

SECTION:element-tensor_demux

A Demuxer that demux tensors stream to tensor(s) stream for NN frameworks. The output is always in the format of other/tensor or other/tensors.

<refsect2> <title>Example launch line</title> |[gst-launch-1.0 tensor_mux name=mux ! tensor_demux name=demux \ filesrc location=testcase01_RGB_100x100.png ! pngdec ! videoscale ! imagefreeze ! videoconvert ! video/x-raw,format=RGB,width=100,height=100,framerate=0/1 ! tensor_converter ! mux.sink_↩
0 \ filesrc location=testcase01_RGB_100x100.png ! pngdec ! videoscale ! imagefreeze ! videoconvert ! video/x-raw,format=RGB,width=100,height=100,framerate=0/1 ! tensor_converter ! mux.sink_1 \ filesrc location=testcase01_RGB_100x100.png ! pngdec ! videoscale ! imagefreeze ! videoconvert ! video/x-raw,format=RGB,width=100,height=100,framerate=0/1 ! tensor_converter ! mux.sink_2 \ demux.src_0 ! queue ! filesink location=demux00.log \ demux.src_1 ! queue ! filesink location=demux01.log \ demux.src_2 ! queue ! filesink location=demux02.log]|

|[gst-launch-1.0 tensor_mux name=mux ! tensor_demux name=demux tensorpick=0,1:2,2+0 \ ... (tensor 0) ! mux.sink_0 \ ... (tensor 1) ! mux.sink_1 \ ... (tensor 2) ! mux.sink_2 \ demux.src_0 ! (tensor 0) ... demux.src_1 ! (tensor 1,2) ... demux.src_2 ! (tensor 2,0) ...]|

</refsect2>

9.38.4.3 gst_tensor_demux_chain()

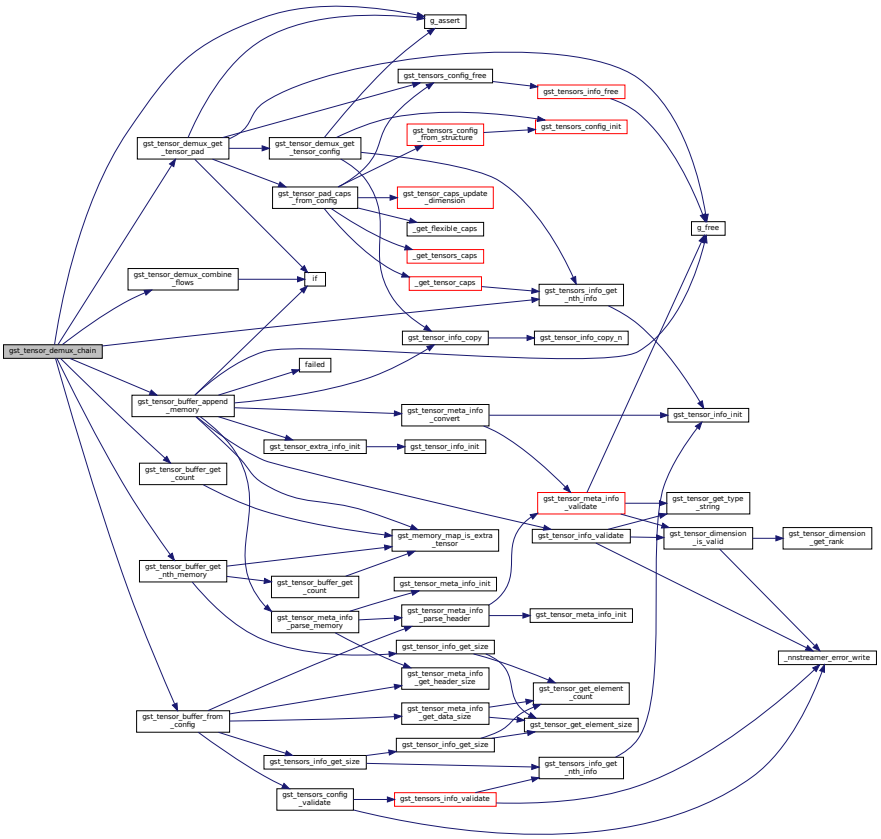
```
static GstFlowReturn gst_tensor_demux_chain (
            GstPad * pad,
            GstObject * parent,
            GstBuffer * buf )  [static]
```

chain function for sink (gst element vmethod)

The number of tensors in the buffer: The number of tensors from caps and gst-buffer should be same when incoming buffer is static tensor. If given buffer is flexible tensor, we cannot get exact number of tensors from config.

Definition at line 476 of file gsttensor_demux.c.

Here is the call graph for this function:



Here is the caller graph for this function:

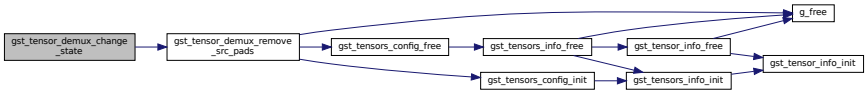
9.38.4.4 gst_tensor_demux_change_state()

```
static GstStateChangeReturn gst_tensor_demux_change_state (
            GstElement * element,
            GstStateChange transition )  [static]
```

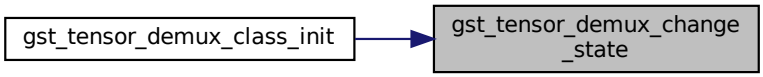
change state (gst element vmethod)

Definition at line 586 of file gsttensor_demux.c.

Here is the call graph for this function:



Here is the caller graph for this function:



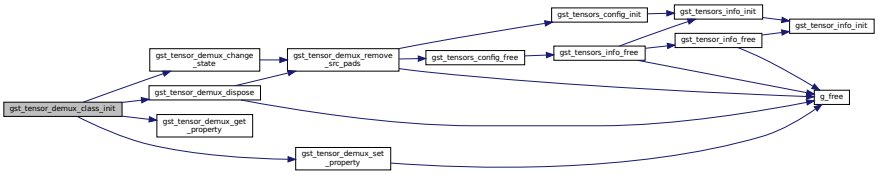
9.38.4.5 gst_tensor_demux_class_init()

```
static void gst_tensor_demux_class_init (
            GstTensorDemuxClass * klass )  [static]
```

initialize the tensor_demux's class

Definition at line 127 of file gsttensor_demux.c.

Here is the call graph for this function:



9.38.4.6 gst_tensor_demux_combine_flows()

```
static GstFlowReturn gst_tensor_demux_combine_flows (
            GstTensorDemux * tensor_demux,
            GstTensorPad * pad,
            GstFlowReturn ret )  [static]
```

Check the status among sources in demux.

Parameters

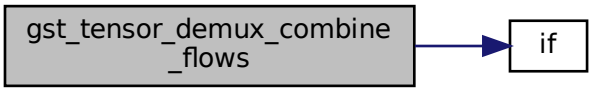
tensor_demux	TensorDemux Object
TensorPad	Tensorpad
ret	return status of current pad

Returns

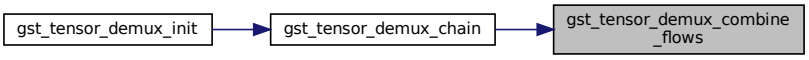
return status after check sources

Definition at line 453 of file gsttensor_demux.c.

Here is the call graph for this function:



Here is the caller graph for this function:



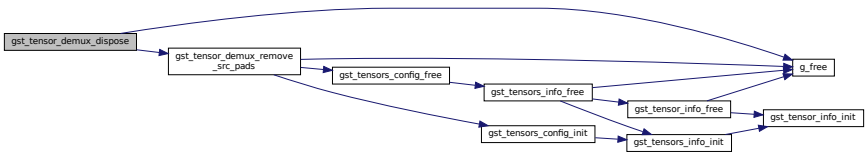
9.38.4.7 gst_tensor_demux_dispose()

```
static void gst_tensor_demux_dispose (
            GObject * object )  [static]
```

dispose function for tensor demux (gst element vmethod)

Definition at line 219 of file gsttensor_demux.c.

Here is the call graph for this function:



Here is the caller graph for this function:

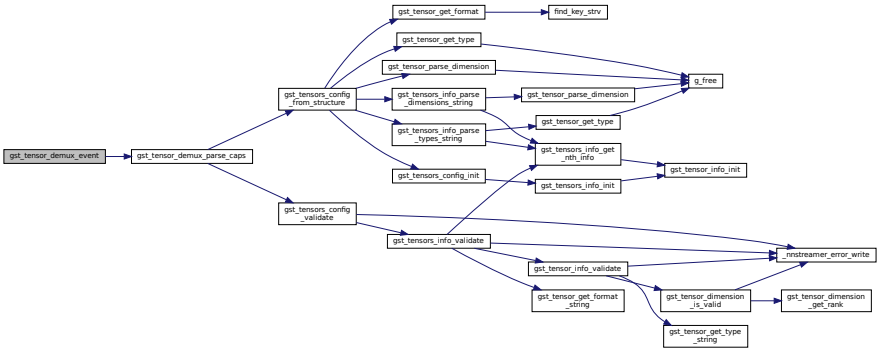
9.38.4.8 gst_tensor_demux_event()

```
static gboolean gst_tensor_demux_event (
            GstPad * pad,
            GstObject * parent,
            GstEvent * event )  [static]
```

event function for sink (gst element vmethod)

Definition at line 252 of file gsttensor_demux.c.

Here is the call graph for this function:



Here is the caller graph for this function:

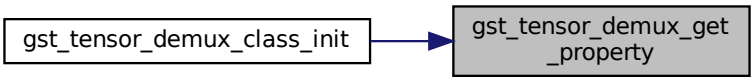
9.38.4.9 gst_tensor_demux_get_property()

```
static void gst_tensor_demux_get_property (
             GObject * object,
             guint prop_id,
             GValue * value,
             GParamSpec * pspec )  [static]
```

Get property (gst element vmethod)

Definition at line 652 of file gsttensor_demux.c.

Here is the caller graph for this function:



9.38.4.10 gst_tensor_demux_get_tensor_config()

```
static gboolean gst_tensor_demux_get_tensor_config (
             GstTensorDemux * tensor_demux,
             GstTensorsConfig * config,
             const guint nth,
             const guint total )  [static]
```

Get tensor config info from configured tensors.

Parameters

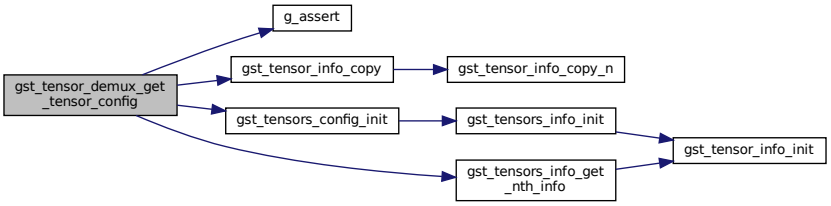
tensor_demux	"this" pointer
config	tensor config to be filled
nth	source ordering
total	number of tensors

Returns

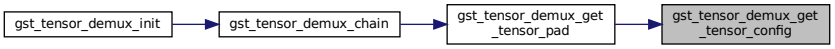
TRUE if successfully configured

Definition at line 290 of file gsttensor_demux.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.38.4.11 gst_tensor_demux_get_tensor_pad()

```
static GstTensorPad* gst_tensor_demux_get_tensor_pad (
            GstTensorDemux * tensor_demux,
            gboolean * created,
            const guint nth,
            const guint total )  [static]
```

Checking if the source pad is created and if not, create TensorPad.

Parameters

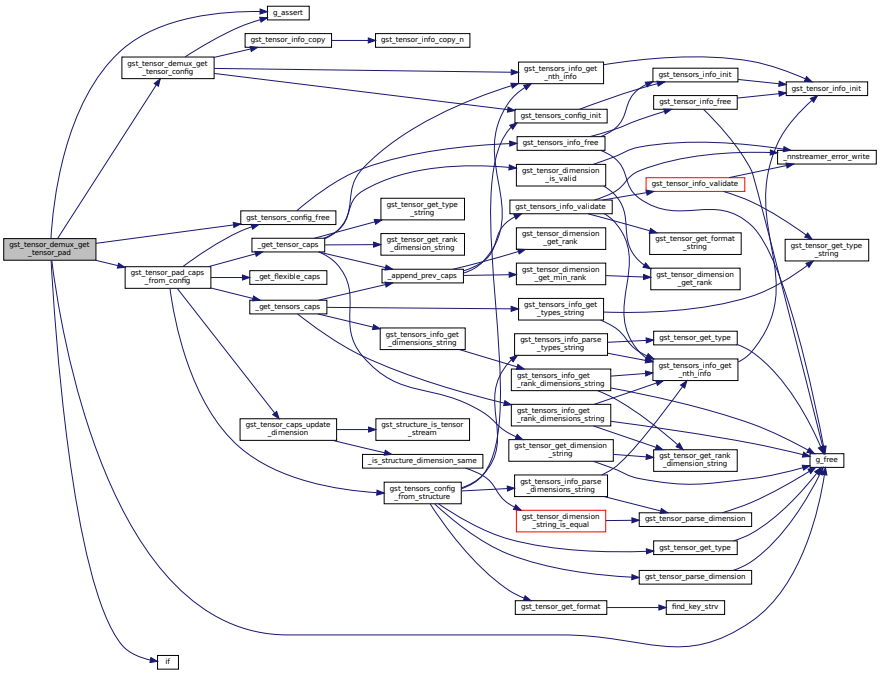
	tensor_demux	TensorDemux Object
out	*created*	will be updated in this function
	nth	source ordering
	total	number of tensors

Returns

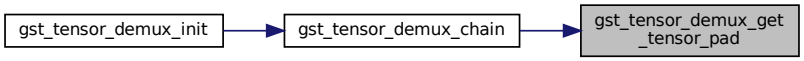
TensorPad if pad is already created, then return created pad. If not return new pad after creation.

Definition at line 349 of file gsttensor_demux.c.

Here is the call graph for this function:



Here is the caller graph for this function:



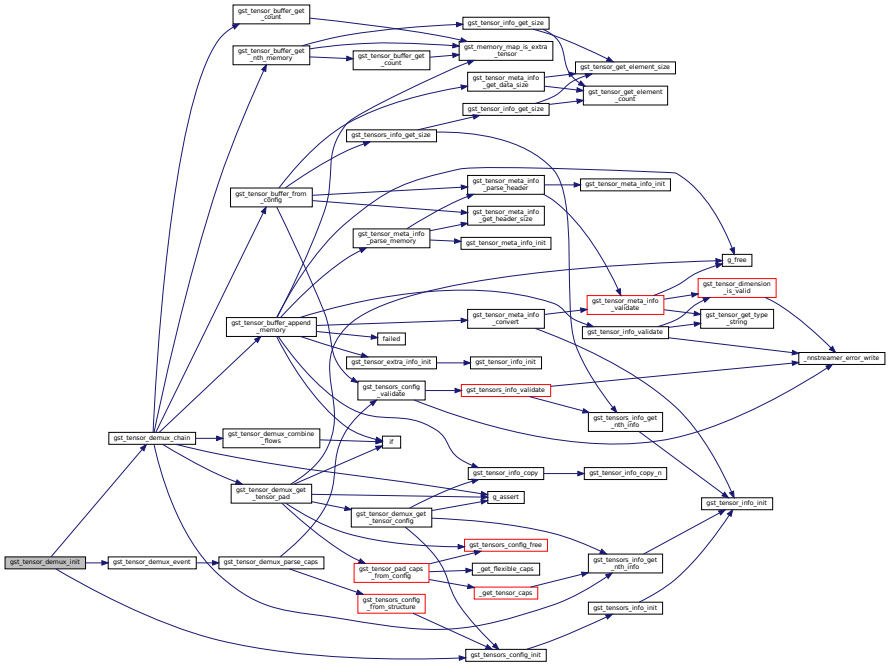
9.38.4.12 gst_tensor_demux_init()

```
static void gst_tensor_demux_init (
            GstTensorDemux * tensor_demux )  [static]
```

initialize the new element instantiate pads and add them to element set pad callback functions initialize instance structure

Definition at line 175 of file gsttensor_demux.c.

Here is the call graph for this function:



9.38.4.13 gst_tensor_demux_parse_caps()

```
static gboolean gst_tensor_demux_parse_caps (
            GstTensorDemux * tensor_demux,
            GstCaps * caps )  [static]
```

Parse caps and configure tensors info.

Parameters

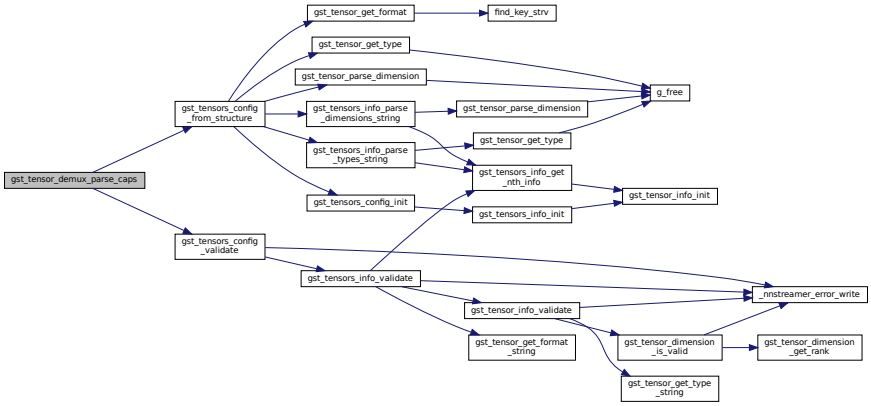
tensor_demux	GstTensorDemux object
caps	incoming capability

Returns

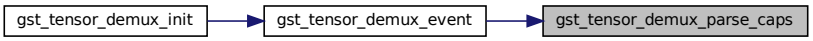
TRUE/FALSE (if successfully configured, return TRUE)

Definition at line 235 of file gsttensor_demux.c.

Here is the call graph for this function:



Here is the caller graph for this function:



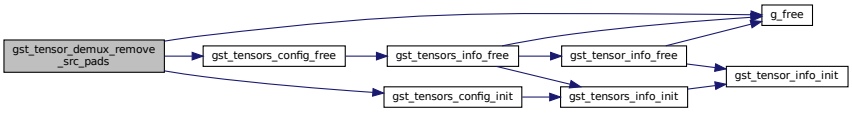
9.38.4.14 gst_tensor_demux_remove_src_pads()

```
static void gst_tensor_demux_remove_src_pads (
            GstTensorDemux * tensor_demux )  [static]
```

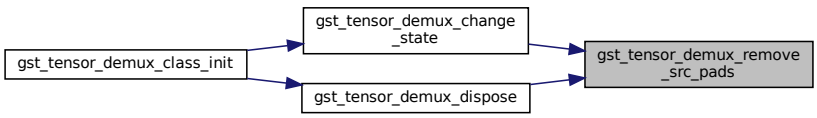
function to remove srcpad list

Definition at line 199 of file gsttensor_demux.c.

Here is the call graph for this function:



Here is the caller graph for this function:



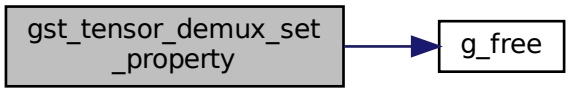
9.38.4.15 gst_tensor_demux_set_property()

```
static void gst_tensor_demux_set_property (
             GObject * object,
             guint prop_id,
             const GValue * value,
             GParamSpec * pspec )  [static]
```

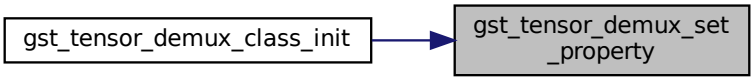
Get property (gst element vmethod)

Definition at line 615 of file gsttensor_demux.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.38.5 Variable Documentation

9.38.5.1 sink_templ

```
GstStaticPadTemplate sink_templ  [static]
```

Initial value:

```
= GST_STATIC_PAD_TEMPLATE ("sink",
    GST_PAD_SINK,
    GST_PAD_ALWAYS,
    )
```

Definition at line 102 of file gsttensor_demux.c.

9.38.5.2 src_templ

```
GstStaticPadTemplate src_templ  [static]
```

Initial value:

```
= GST_STATIC_PAD_TEMPLATE ("src_%u",
    GST_PAD_SRC,
    GST_PAD_SOMETIMES,
    )
```

the capabilities of the inputs and outputs. describe the real formats here.

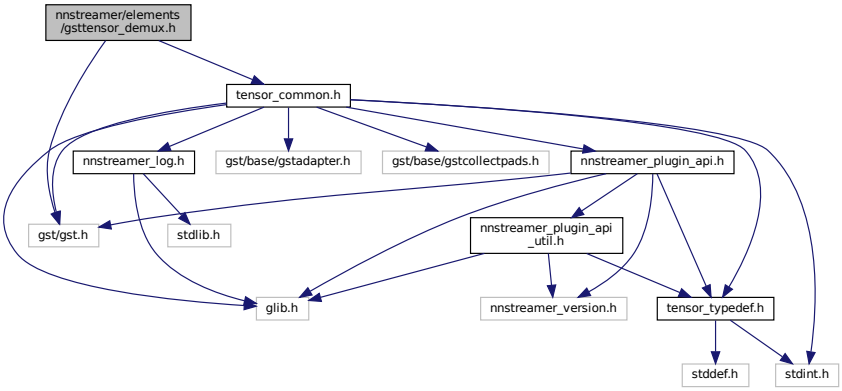
Definition at line 96 of file gsttensor_demux.c.

9.39 nnstreamer/elements/gsttensor_demux.h File Reference

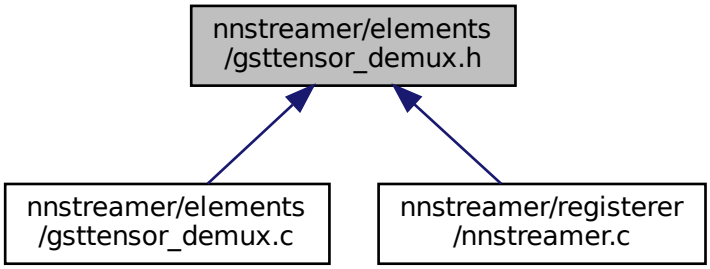
GStreamer plugin to demux tensors (as a filter for other general neural network filters)

```
#include <gst/gst.h>
#include <tensor_common.h>
```

Include dependency graph for gsttensor_demux.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct _GstTensorDemux

  Tensor Muxer data structure.
- struct _GstTensorDemuxClass

  GstTensorDeMuxClass inherits GstElementClass.

Macros

- #define GST_TYPE_TENSOR_DEMUX (gst_tensor_demux_get_type ())
- #define GST_TENSOR_DEMUX(obj) (G_TYPE_CHECK_INSTANCE_CAST ((obj), GST_TYPE_TENSOR_DEMUX, GstTensorDemux))
- #define GST_TENSOR_DEMUX_CLASS(klass) (G_TYPE_CHECK_CLASS_CAST ((klass), GST_TYPE_TENSOR_DEMUX, GstTensorDemuxClass))
- #define GST_TENSOR_DEMUX_GET_CLASS(obj) (G_TYPE_INSTANCE_GET_CLASS ((obj), GST_TYPE_TENSOR_DEMU GstTensorDemuxClass))
- #define GST_IS_TENSOR_DEMUX(obj) (G_TYPE_CHECK_INSTANCE_TYPE((obj),GST_TYPE_TENSOR_DEMUX))
- #define GST_IS_TENSOR_DEMUX_CLASS(klass) (G_TYPE_CHECK_CLASS_TYPE((klass),GST_TYPE_TENSOR_DEMUX
- #define GST_TENSOR_DEMUX_CAST(obj) ((GstTensorDemux*)(obj))

Typedefs

- typedef struct _GstTensorDemux GstTensorDemux
- typedef struct _GstTensorDemuxClass GstTensorDemuxClass

Functions

- GType gst_tensor_demux_get_type (void)

  Get Type function required for gst elements.

9.39.1 Detailed Description

GStreamer plugin to demux tensors (as a filter for other general neural network filters)

GStreamer Copyright (C) 2005 Thomas Vander Stichele thomas@apestaart.org Copyright (C) 2005 Ronald S. Bultje rbultje@ronald.bitfreak.net Copyright (C) 2018 Jijoong Moon jijoong.moon@samsung.com

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Library General Public License as published by the Free Software Foundation; version 2.1 of the License.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Library General Public License for more details.

Date

03 July 2018

See also

https://github.com/nnstreamer/nnstreamer

Author

Jijoong Moon jijoong.moon@samsung.com

Bug No known bugs except for NYI items

9.39.2 Macro Definition Documentation

9.39.2.1 GST_IS_TENSOR_DEMUX

```
#define GST_IS_TENSOR_DEMUX(
              obj ) (G_TYPE_CHECK_INSTANCE_TYPE((obj),GST_TYPE_TENSOR_DEMUX))
```

Definition at line 40 of file gsttensor_demux.h.

9.39.2.2 GST_IS_TENSOR_DEMUX_CLASS

```
#define GST_IS_TENSOR_DEMUX_CLASS(
              klass ) (G_TYPE_CHECK_CLASS_TYPE((klass),GST_TYPE_TENSOR_DEMUX))
```

Definition at line 41 of file gsttensor_demux.h.

9.39.2.3 GST_TENSOR_DEMUX

```
#define GST_TENSOR_DEMUX(
              obj ) (G_TYPE_CHECK_INSTANCE_CAST ((obj), GST_TYPE_TENSOR_DEMUX, GstTensorDemux))
```

Definition at line 37 of file gsttensor_demux.h.

9.39.2.4 GST_TENSOR_DEMUX_CAST

```
#define GST_TENSOR_DEMUX_CAST(
              obj ) ((GstTensorDemux*)(obj))
```

Definition at line 42 of file gsttensor_demux.h.

9.39.2.5 GST_TENSOR_DEMUX_CLASS

```
#define GST_TENSOR_DEMUX_CLASS(
              klass ) (G_TYPE_CHECK_CLASS_CAST ((klass), GST_TYPE_TENSOR_DEMUX, GstTensorDemuxClass))
```

Definition at line 38 of file gsttensor_demux.h.

9.39.2.6 GST_TENSOR_DEMUX_GET_CLASS

```
#define GST_TENSOR_DEMUX_GET_CLASS(
              obj ) (G_TYPE_INSTANCE_GET_CLASS ((obj), GST_TYPE_TENSOR_DEMUX, GstTensorDemuxClass))
```

Definition at line 39 of file gsttensor_demux.h.

9.39.2.7 GST_TYPE_TENSOR_DEMUX

```
#define GST_TYPE_TENSOR_DEMUX (gst_tensor_demux_get_type ())
```

Definition at line 36 of file gsttensor_demux.h.

9.39.3 Typedef Documentation

9.39.3.1 GstTensorDemux

```
typedef struct _GstTensorDemux GstTensorDemux
```

Definition at line 44 of file gsttensor_demux.h.

9.39.3.2 GstTensorDemuxClass

```
typedef struct _GstTensorDemuxClass GstTensorDemuxClass
```

Definition at line 45 of file gsttensor_demux.h.

9.39.4 Function Documentation

9.39.4.1 gst_tensor_demux_get_type()

```
GType gst_tensor_demux_get_type (
             void  )
```

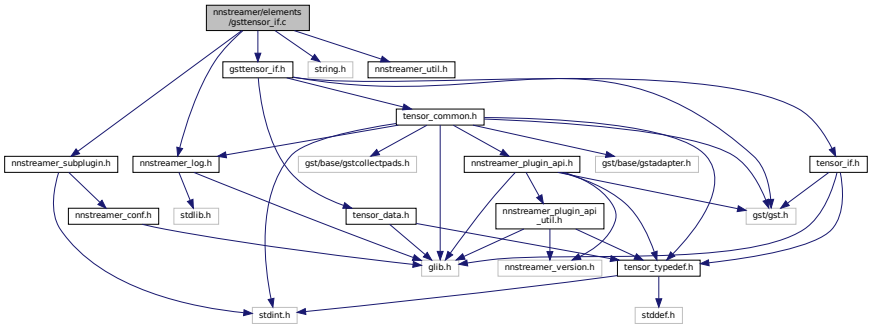
Get Type function required for gst elements.

9.40 nnstreamer/elements/gsttensor_if.c File Reference

GStreamer plugin to control flow based on tensor values.

```
#include <nnstreamer_log.h>
#include <string.h>
#include <nnstreamer_subplugin.h>
#include <nnstreamer_util.h>
#include "gsttensor_if.h"
```

Include dependency graph for gsttensor_if.c:



Macros

- #define DBG (!tensor_if->silent)

  Macro for debug mode.
- #define GST_CAT_DEFAULT gst_tensor_if_debug
- #define CAPS_STRING GST_TENSOR_CAP_DEFAULT "; " GST_TENSORS_CAP_DEFAULT
- #define gst_tensor_if_parent_class parent_class
- #define GST_TYPE_TENSOR_IF_CV (gst_tensor_if_cv_get_type ())
- #define GST_TYPE_TENSOR_IF_OP (gst_tensor_if_op_get_type ())
- #define GST_TYPE_TENSOR_IF_ACT (gst_tensor_if_act_get_type ())
- #define operator_func(cv, t, op, sv1, sv2, ret)

  Macro for operator function.

Enumerations

- enum {
  PROP_0, PROP_SILENT, PROP_CV, PROP_CV_OPTION,
  PROP_OP, PROP_SV, PROP_THEN, PROP_THEN_OPTION,
  PROP_ELSE, PROP_ELSE_OPTION }

  tensor_if properties

Functions

- GST_DEBUG_CATEGORY_STATIC (gst_tensor_if_debug)
- G_DEFINE_TYPE (GstTensorIf, gst_tensor_if, GST_TYPE_ELEMENT)
- static void gst_tensor_if_set_property (GObject *object, guint prop_id, const GValue *value, GParamSpec *pspec)

  Setter for tensor_if properties.
- static void gst_tensor_if_get_property (GObject *object, guint prop_id, GValue *value, GParamSpec *pspec)

  Getter for tensor_if properties.
- static GstFlowReturn gst_tensor_if_chain (GstPad *pad, GstObject *parent, GstBuffer *buf)

  chain function for sink (gst element vmethod)
- static gboolean gst_tensor_if_event (GstPad *pad, GstObject *parent, GstEvent *event)

  event function for sink (gst element vmethod)
- static void gst_tensor_if_dispose (GObject *object)

  dispose function for tensor if (gst element vmethod)
- static void gst_tensor_if_install_properties (GObjectClass *gobject_class)

  Installs all the properties for tensor_if.
- static GType gst_tensor_if_cv_get_type (void)

  A private function to register GEnumValue array for the 'compared-value' property to a GType and return it.
- static GType gst_tensor_if_op_get_type (void)

  A private function to register GEnumValue array for the 'operator' property to a GType and return it.
- static GType gst_tensor_if_act_get_type (void)

  A private function to register GEnumValue array for the 'then' and 'else' properties to a GType and return it.
- static void gst_tensor_if_class_init (GstTensorIfClass *klass)

  initialize the tensor_if's class (GST Standard)
- static void gst_tensor_if_init (GstTensorIf *tensor_if)

  initialize the new element (GST Standard) instantiate pads and add them to element set pad callback functions initialize instance structure
- static void gst_tensor_if_remove_src_pads (GstTensorIf *tensor_if)

  function to remove srcpad list
- static void gst_tensor_if_set_property_glist (const GValue *value, GList **prop_list, const gchar *delimiters)

  Convert GValue to GList according to delimiters.
- static void gst_tensor_if_set_property_cv_option (const GValue *value, GList **prop_list)

  Convert GValue to GList for cv option.
- static void gst_tensor_if_set_property_supplied_value (const GValue *value, tensor_if_sv_s *sv, const gchar *delimiters)

  Convert GValue to GList according to delimiters.
- static void gst_tensor_if_configure_custom_prop (GstTensorIf *self)

  Set custom compared value property.
- static void gst_tensor_if_property_to_string (GValue *value, GList *prop_list, guint prop_id)

  Convert GList to GValue.
- static void gst_tensor_if_get_property_supplied_value (GValue *value, tensor_if_sv_s *sv)

  Convert GValue to supplied value according to delimiters.
- static gboolean gst_tensor_if_parse_caps (GstTensorIf *tensor_if, GstCaps *caps)

  Parse caps and configure tensors info.
- static GstTensorPad * gst_tensor_if_get_tensor_pad (GstTensorIf *tensor_if, GstTensorsConfig *config, gboolean *created, guint nth)

  Checking if the source pad is created and if not, create TensorPad.
- static GstFlowReturn gst_tensor_if_combine_flows (GstTensorIf *tensor_if, GstTensorPad *pad, GstFlow↩ Return ret)

  Check the status among sources in if.
- switch (cv->type)

- static gboolean gst_tensor_if_get_tensor_average (GstTensorIf *tensor_if, GstBuffer *buf, tensor_data_s *cv, guint nth)

  Calculate average value of the nth tensor.
- static gboolean gst_tensor_if_calculate_cv (GstTensorIf *tensor_if, GstBuffer *buf, tensor_data_s *cv)

  Calculate compared value.
- int nnstreamer_if_custom_register (const gchar *name, tensor_if_custom func, void *data)

  Registers a callback for tensor_if custom condition.
- int nnstreamer_if_custom_unregister (const gchar *name)

  Unregisters a callback for tensor_if custom condition.
- static gboolean gst_tensor_if_check_condition (GstTensorIf *tensor_if, GstBuffer *buf, gboolean *result)

  Determining whether a given condition is true or false.

Variables

- static GstStaticPadTemplate sink_factory

  The capabilities of the inputs.
- static GstStaticPadTemplate src_factory

  The capabilities of the outputs.
- case TIFOP_NE

  *t == sv1._##t) ? TRUE : FALSE; break; *
- case tensor_data_s * cv
- case tensor_data_s gboolean * result
- tensor_data_s svtc_1
- tensor_data_s svtc_2
- svtc_1 type = tensor_if->sv->type
- svtc_1 data = tensor_if->sv->data[0]
- return TRUE

9.40.1 Detailed Description

GStreamer plugin to control flow based on tensor values.

GStreamer/NNStreamer Tensor-IF Copyright (C) 2020 MyungJoo Ham myungjoo.ham@samsung.com

Date

08 April 2020

See also

https://github.com/nnstreamer/nnstreamer

Author

MyungJoo Ham myungjoo.ham@samsung.com

Bug No known bugs except for NYI items

9.40.2 Macro Definition Documentation

9.40.2.1 CAPS_STRING

```
#define CAPS_STRING GST_TENSOR_CAP_DEFAULT "; " GST_TENSORS_CAP_DEFAULT
```

Definition at line 105 of file gsttensor_if.c.

9.40.2.2 DBG

```
#define DBG (!tensor_if->silent)
```

Macro for debug mode.

SECTION:element-tensor_if

A filter that controls its src-pad based on the values (other/tensor(s)) of its sink-pad. For example, you may skip frames if there is no object detected with high confidence.

The format of statement with tensor-if is: if (Compared_Value OPERATOR Supplied_Value(s))) then THEN else ELSE Compared_Value and Supplied_Value are the operands. Compared_Value is a value from input tensor(s). Supplied_Value is a value from tensor-if properties.

If the given if-condition is simple enough (e.g., if a specific element is between a given range in a tensor frame), it can be expressed as: <refsect2> <title>Example launch line with simple if condition</title> gst-launch ... (some tensor stream) ! tensor_if name=tif compared-value=A_VALUE compared-value-option=3:4:2:5,0 operator=RANGE↩_INCLUSIVE supplied-value=10,100 then=PASSTHROUGH else=TENSORPICK else-option=1 tif.src_0 ! queue ! (tensor(s) stream for TRUE action) ... tif.src_1 ! queue ! (tensor(s) stream for FALSE action) ... </refsect2>

However, if the if-condition is complex and cannot be expressed with tensor-if expressions, you may create a corresponding custom filter with tensor-filter, whose output is other/tensors with an additional tensor that is "1:1:1:1, uint8", which is 1 (true) or 0 (false) as the first tensor of other/tensors and the input tensor/tensors. Then, you can create a pipeline as follows: <refsect2> <title>Example launch line with complex if condition</title> gst-launch ... (some tensor stream) ! tensor_filter framework=custom name=your_code.so ! tensor_if compared-value=A_VALUE compared-value-option=0:0:0:0,0 # 1st tensor's [0][0][0][0]. operator=EQ supplied-value=1 then=PASSTHROUGH # or whatsoever you want else=SKIP # or whatsoever you want ! tensor_demux name=d d.src_0 ! queue ! fakesink # throw away the 1/0 value. d.src_1 ! queue ! do whatever you want here... ... </refsect2>

Definition at line 82 of file gsttensor_if.c.

9.40.2.3 GST_CAT_DEFAULT

```
#define GST_CAT_DEFAULT gst_tensor_if_debug
```

Definition at line 103 of file gsttensor_if.c.

9.40.2.4 gst_tensor_if_parent_class

```
#define gst_tensor_if_parent_class parent_class
```

Definition at line 122 of file gsttensor_if.c.

9.40.2.5 GST_TYPE_TENSOR_IF_ACT

```
#define GST_TYPE_TENSOR_IF_ACT (gst_tensor_if_act_get_type ())
```

Definition at line 195 of file gsttensor_if.c.

9.40.2.6 GST_TYPE_TENSOR_IF_CV

```
#define GST_TYPE_TENSOR_IF_CV (gst_tensor_if_cv_get_type ())
```

Definition at line 139 of file gsttensor_if.c.

9.40.2.7 GST_TYPE_TENSOR_IF_OP

```
#define GST_TYPE_TENSOR_IF_OP (gst_tensor_if_op_get_type ())
```

Definition at line 163 of file gsttensor_if.c.

9.40.2.8 operator_func

```
#define operator_func(
              cv,
              t,
              op,
              sv1,
              sv2,
              ret )
```

Value:

```
  do { \
  switch (op) { \
    case TIFOP_EQ: ret = (cv._
```

Macro for operator function.

Definition at line 813 of file gsttensor_if.c.

9.40.3 Enumeration Type Documentation

9.40.3.1 anonymous enum

```
anonymous enum
```

tensor_if properties

Enumerator

PROP_0	
PROP_SILENT	
PROP_CV	Compared Value, operand 1 (from input tensor(s))
PROP_CV_OPTION	Compared Value Option
PROP_OP	Operator
PROP_SV	Supplied Value, operand 2 (from the properties)
PROP_THEN	Action if it is TRUE
PROP_THEN_OPTION	Option for TRUE Action
PROP_ELSE	Action if it is FALSE
PROP_ELSE_OPTION	Option for FALSE Action

Definition at line 88 of file gsttensor_if.c.

9.40.4 Function Documentation

9.40.4.1 G_DEFINE_TYPE()

```
G_DEFINE_TYPE (
            GstTensorIf ,
            gst_tensor_if ,
            GST_TYPE_ELEMENT  )
```

9.40.4.2 GST_DEBUG_CATEGORY_STATIC()

```
GST_DEBUG_CATEGORY_STATIC (
            gst_tensor_if_debug  )
```

9.40.4.3 gst_tensor_if_act_get_type()

```
static GType gst_tensor_if_act_get_type (
            void  )  [static]
```

A private function to register GEnumValue array for the 'then' and 'else' properties to a GType and return it.

Definition at line 201 of file gsttensor_if.c.

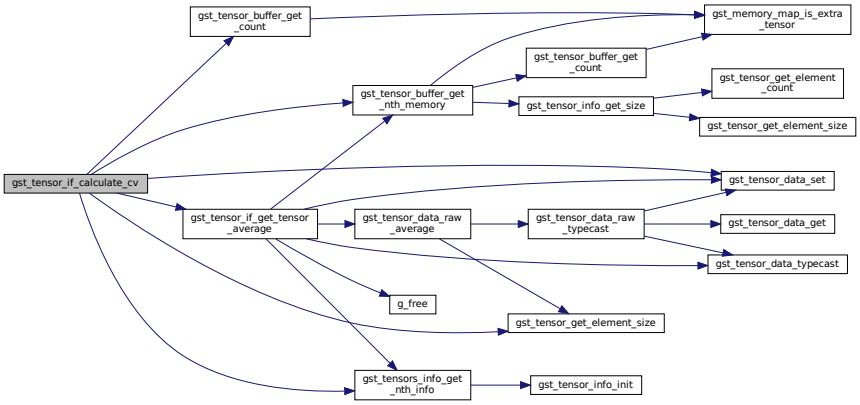
9.40.4.4 gst_tensor_if_calculate_cv()

```
static gboolean gst_tensor_if_calculate_cv (
            GstTensorIf * tensor_if,
            GstBuffer * buf,
            tensor_data_s * cv )  [static]
```

Calculate compared value.

Definition at line 939 of file gsttensor_if.c.

Here is the call graph for this function:



Here is the caller graph for this function:

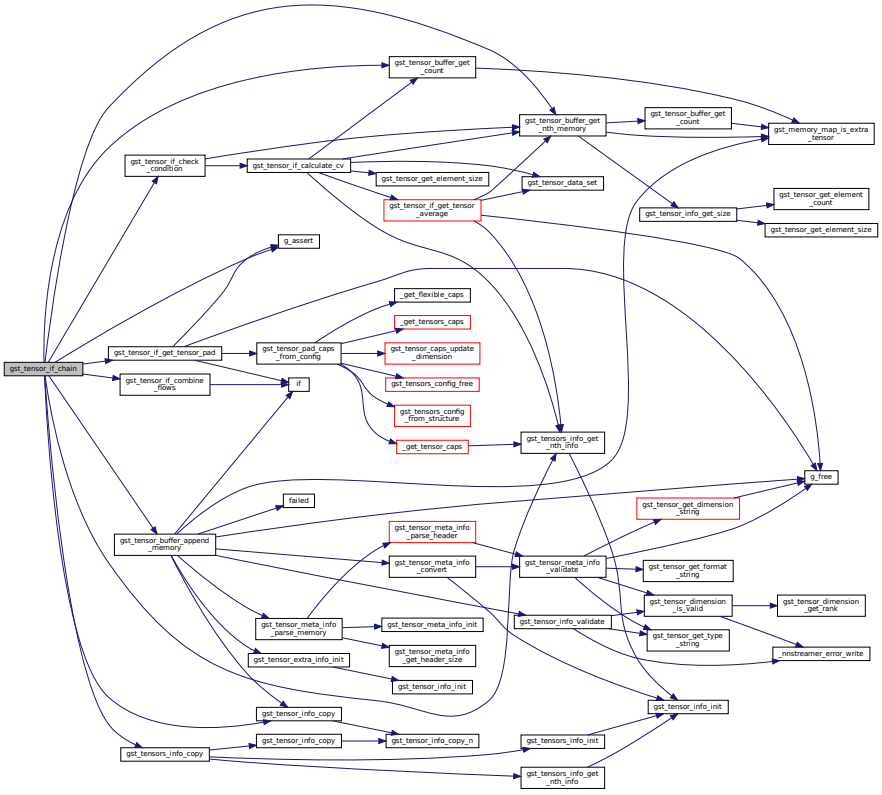
9.40.4.5 gst_tensor_if_chain()

```
static GstFlowReturn gst_tensor_if_chain (
            GstPad * pad,
            GstObject * parent,
            GstBuffer * buf )  [static]
```

chain function for sink (gst element vmethod)

Definition at line 1125 of file gsttensor_if.c.

Here is the call graph for this function:



Here is the caller graph for this function:

9.40.4.6 gst_tensor_if_check_condition()

```
static gboolean gst_tensor_if_check_condition (
            GstTensorIf * tensor_if,
            GstBuffer * buf,
            gboolean * result )  [static]
```

Determining whether a given condition is true or false.

Parameters

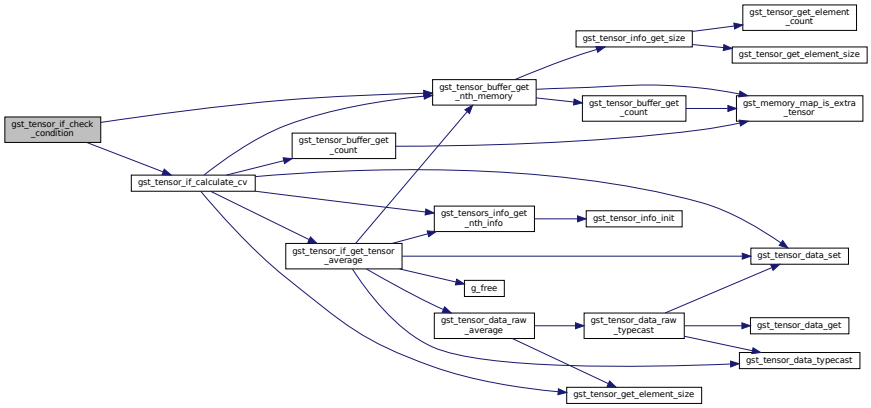
tensor_if	TensorIf Object
buf	gstbuffer from sink pad

Returns

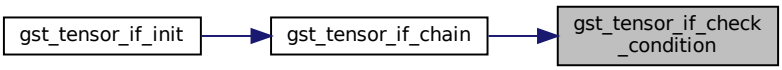
return TRUE if no error

Definition at line 1070 of file gsttensor_if.c.

Here is the call graph for this function:



Here is the caller graph for this function:



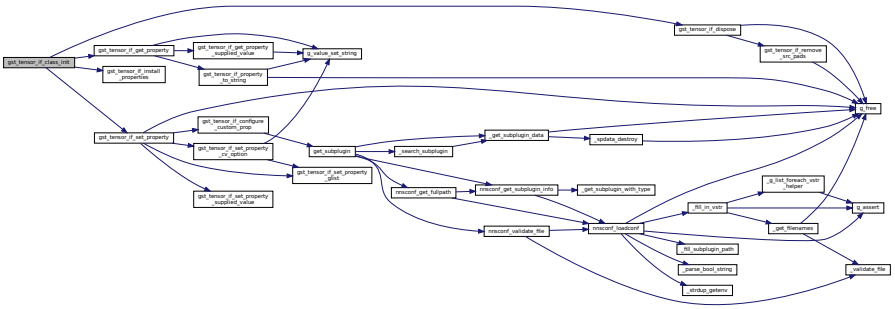
9.40.4.7 gst_tensor_if_class_init()

```
static void gst_tensor_if_class_init (
            GstTensorIfClass * klass )  [static]
```

initialize the tensor_if's class (GST Standard)

Definition at line 222 of file gsttensor_if.c.

Here is the call graph for this function:



9.40.4.8 gst_tensor_if_combine_flows()

```
static GstFlowReturn gst_tensor_if_combine_flows (
            GstTensorIf * tensor_if,
            GstTensorPad * pad,
            GstFlowReturn ret )  [static]
```

Check the status among sources in if.

Parameters

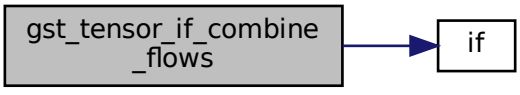
tensor_if	TensorIf Object
TensorPad	Tensorpad
ret	return status of current pad

Returns

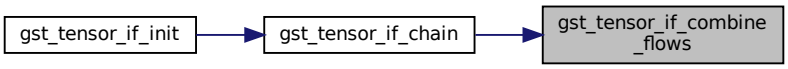
return status after check sources

Definition at line 791 of file gsttensor_if.c.

Here is the call graph for this function:



Here is the caller graph for this function:



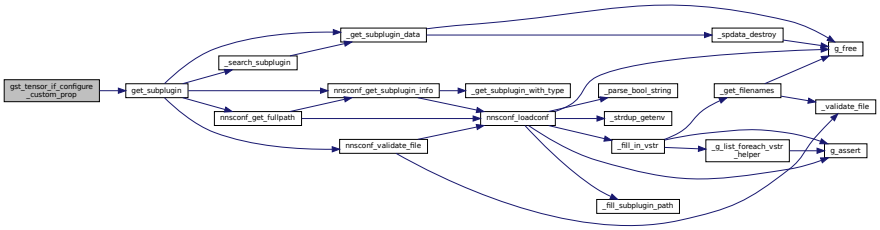
9.40.4.9 gst_tensor_if_configure_custom_prop()

```
static void gst_tensor_if_configure_custom_prop (
            GstTensorIf * self )  [static]
```

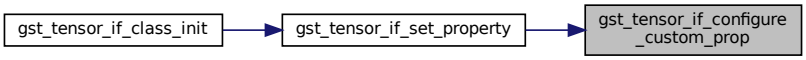
Set custom compared value property.

Definition at line 435 of file gsttensor_if.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.40.4.10 gst_tensor_if_cv_get_type()

```
static GType gst_tensor_if_cv_get_type (
            void  )  [static]
```

A private function to register GEnumValue array for the 'compared-value' property to a GType and return it.

Definition at line 145 of file gsttensor_if.c.

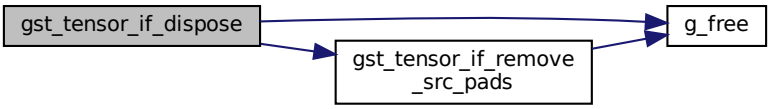
9.40.4.11 gst_tensor_if_dispose()

```
static void gst_tensor_if_dispose (
             GObject * object )  [static]
```

dispose function for tensor if (gst element vmethod)

Definition at line 307 of file gsttensor_if.c.

Here is the call graph for this function:



Here is the caller graph for this function:

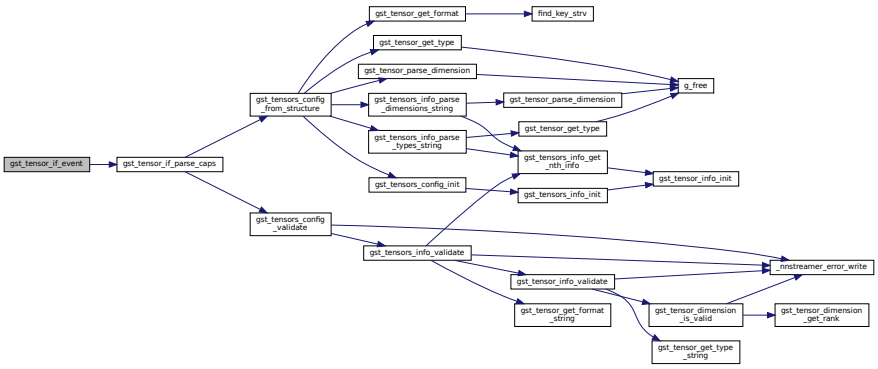
9.40.4.12 gst_tensor_if_event()

```
static gboolean gst_tensor_if_event (
             GstPad * pad,
             GstObject * parent,
             GstEvent * event )  [static]
```

event function for sink (gst element vmethod)

Definition at line 695 of file gsttensor_if.c.

Here is the call graph for this function:



Here is the caller graph for this function:

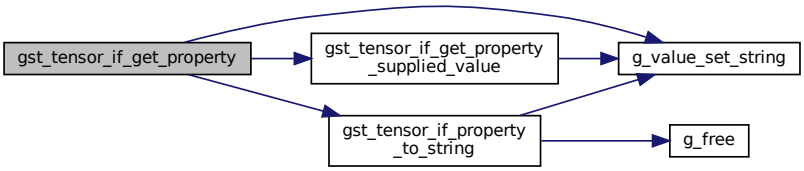
9.40.4.13 gst_tensor_if_get_property()

```
static void gst_tensor_if_get_property (
            GObject * object,
            guint prop_id,
            GValue * value,
            GParamSpec * pspec )  [static]
```

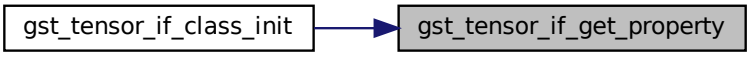
Getter for tensor_if properties.

Definition at line 579 of file gsttensor_if.c.

Here is the call graph for this function:



Here is the caller graph for this function:



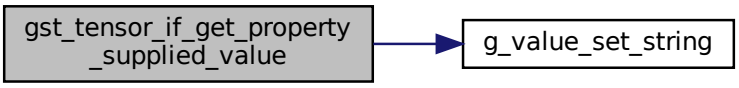
9.40.4.14 gst_tensor_if_get_property_supplied_value()

```
static void gst_tensor_if_get_property_supplied_value (
            GValue * value,
            tensor_if_sv_s * sv )  [static]
```

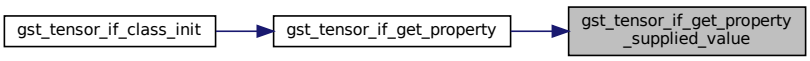
Convert GValue to supplied value according to delimiters.

Definition at line 547 of file gsttensor_if.c.

Here is the call graph for this function:



Here is the caller graph for this function:



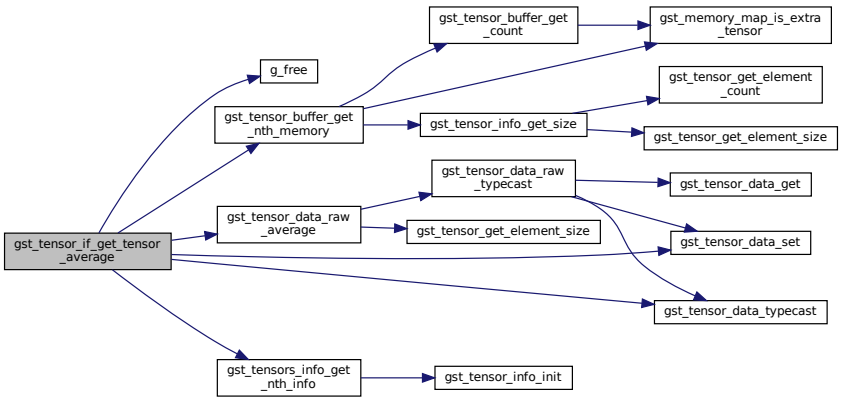
9.40.4.15 gst_tensor_if_get_tensor_average()

```
static gboolean gst_tensor_if_get_tensor_average (
            GstTensorIf * tensor_if,
            GstBuffer * buf,
            tensor_data_s * cv,
            guint nth )  [static]
```

Calculate average value of the nth tensor.

Definition at line 904 of file gsttensor_if.c.

Here is the call graph for this function:



Here is the caller graph for this function:

9.40.4.16 gst_tensor_if_get_tensor_pad()

```
static GstTensorPad* gst_tensor_if_get_tensor_pad (
            GstTensorIf * tensor_if,
            GstTensorsConfig * config,
            gboolean * created,
            guint nth )  [static]
```

Checking if the source pad is created and if not, create TensorPad.

Parameters

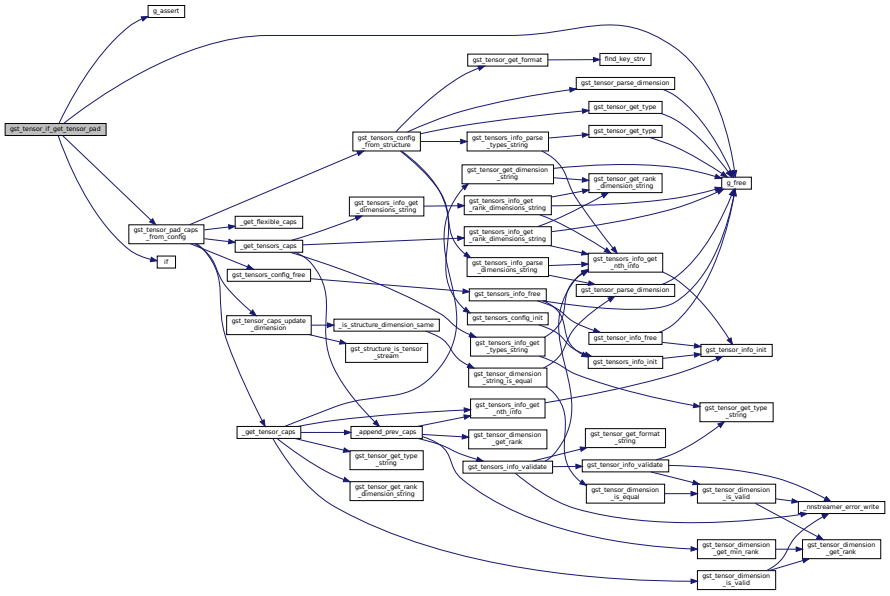
tensor_if	TensorIf Object
config	Tensors Config Data
nth	source ordering

Returns

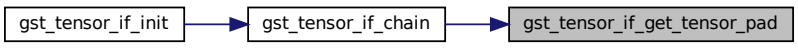
TensorPad if pad is already created, then return created pad. If not return new pad after creation.

Definition at line 727 of file gsttensor_if.c.

Here is the call graph for this function:



Here is the caller graph for this function:



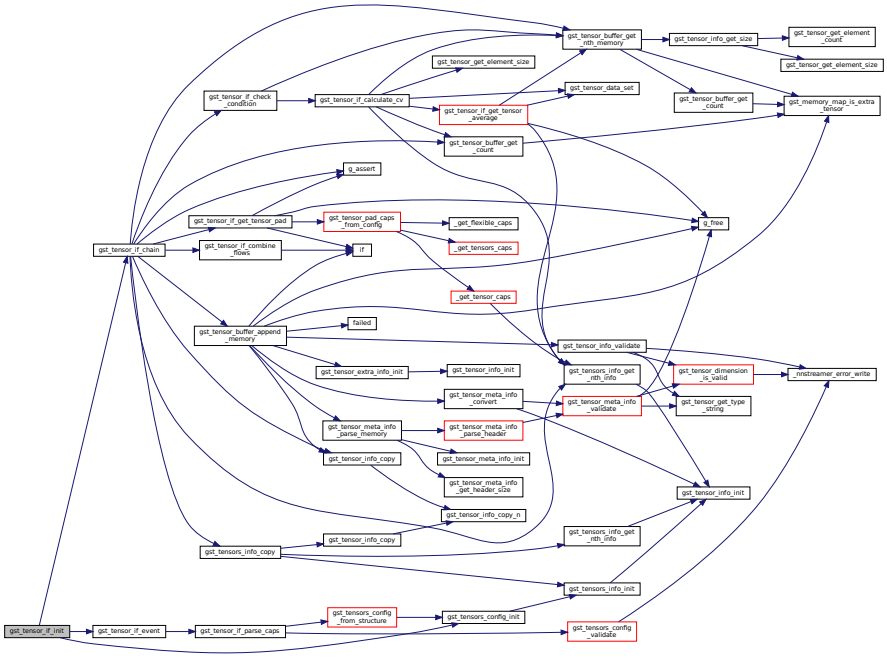
9.40.4.17 gst_tensor_if_init()

```
static void gst_tensor_if_init (
            GstTensorIf * tensor_if )  [static]
```

initialize the new element (GST Standard) instantiate pads and add them to element set pad callback functions initialize instance structure

Definition at line 260 of file gsttensor_if.c.

Here is the call graph for this function:



9.40.4.18 gst_tensor_if_install_properties()

```
static void gst_tensor_if_install_properties (
            GObjectClass * gobject_class )  [static]
```

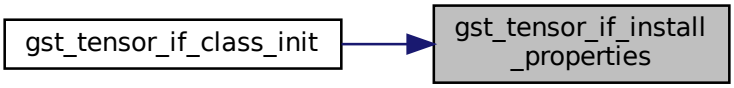
Installs all the properties for tensor_if.

Parameters

in	*gobject_class*	Glib object class whose properties will be set

Definition at line 627 of file gsttensor_if.c.

Here is the caller graph for this function:



9.40.4.19 gst_tensor_if_op_get_type()

```
static GType gst_tensor_if_op_get_type (
            void  )  [static]
```

A private function to register GEnumValue array for the 'operator' property to a GType and return it.

Definition at line 169 of file gsttensor_if.c.

9.40.4.20 gst_tensor_if_parse_caps()

```
static gboolean gst_tensor_if_parse_caps (
            GstTensorIf * tensor_if,
            GstCaps * caps )  [static]
```

Parse caps and configure tensors info.

Parameters

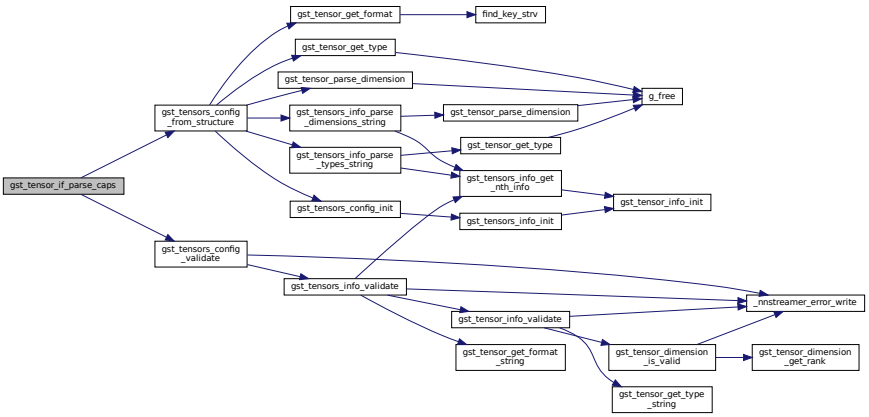
tensor_if	GstTensorIf object
caps	incoming capability

Returns

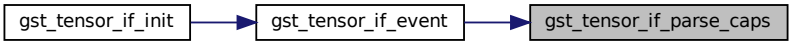
TRUE/FALSE (if successfully configured, return TRUE)

Definition at line 679 of file gsttensor_if.c.

Here is the call graph for this function:



Here is the caller graph for this function:



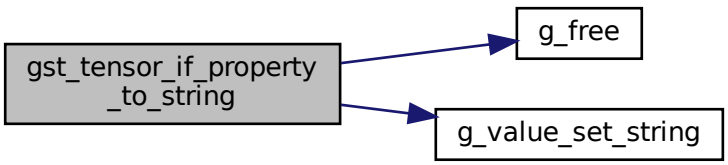
9.40.4.21 gst_tensor_if_property_to_string()

```
static void gst_tensor_if_property_to_string (
            GValue * value,
            GList * prop_list,
            guint prop_id )  [static]
```

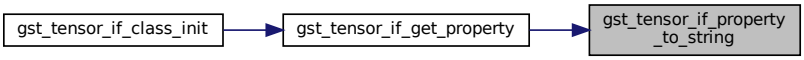
Convert GList to GValue.

Definition at line 503 of file gsttensor_if.c.

Here is the call graph for this function:



Here is the caller graph for this function:



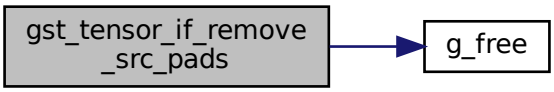
9.40.4.22 gst_tensor_if_remove_src_pads()

```
static void gst_tensor_if_remove_src_pads (
            GstTensorIf * tensor_if )  [static]
```

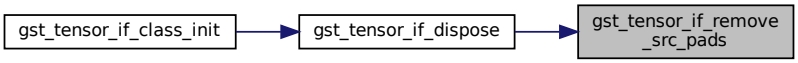
function to remove srcpad list

Definition at line 290 of file gsttensor_if.c.

Here is the call graph for this function:



Here is the caller graph for this function:



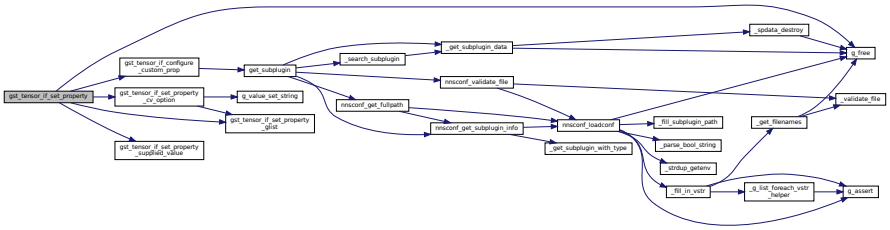
9.40.4.23 gst_tensor_if_set_property()

```
static void gst_tensor_if_set_property (
            GObject * object,
            guint prop_id,
            const GValue * value,
            GParamSpec * pspec )  [static]
```

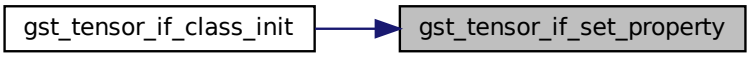
Setter for tensor_if properties.

Definition at line 456 of file gsttensor_if.c.

Here is the call graph for this function:



Here is the caller graph for this function:



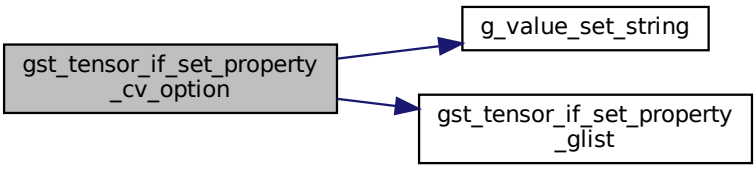
9.40.4.24 gst_tensor_if_set_property_cv_option()

```
static void gst_tensor_if_set_property_cv_option (
            const GValue * value,
            GList ** prop_list )  [static]
```

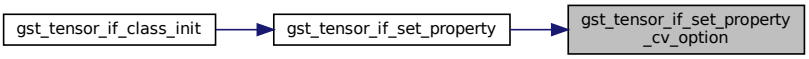
Convert GValue to GList for cv option.

Definition at line 354 of file gsttensor_if.c.

Here is the call graph for this function:



Here is the caller graph for this function:



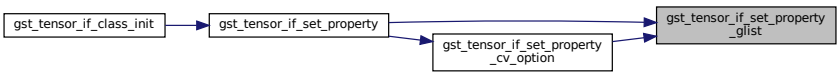
9.40.4.25 gst_tensor_if_set_property_glist()

```
static void gst_tensor_if_set_property_glist (
            const GValue * value,
            GList ** prop_list,
            const gchar * delimiters )  [static]
```

Convert GValue to GList according to delimiters.

Definition at line 328 of file gsttensor_if.c.

Here is the caller graph for this function:



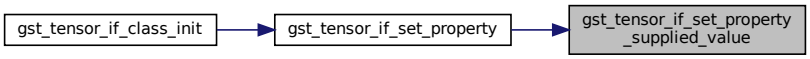
9.40.4.26 gst_tensor_if_set_property_supplied_value()

```
static void gst_tensor_if_set_property_supplied_value (
            const GValue * value,
            tensor_if_sv_s * sv,
            const gchar * delimiters )  [static]
```

Convert GValue to GList according to delimiters.

Definition at line 400 of file gsttensor_if.c.

Here is the caller graph for this function:



9.40.4.27 nnstreamer_if_custom_register()

```
int nnstreamer_if_custom_register (
            const gchar * name,
            tensor_if_custom func,
            void * data )
```

Registers a callback for tensor_if custom condition.

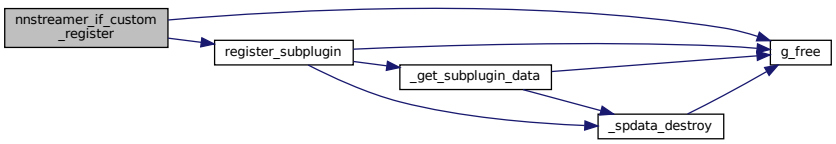
Register the custom callback function.

Returns

0 if success. -ERRNO if error.

Definition at line 1023 of file gsttensor_if.c.

Here is the call graph for this function:



9.40.4.28 nnstreamer_if_custom_unregister()

```
int nnstreamer_if_custom_unregister (
            const gchar * name )
```

Unregisters a callback for tensor_if custom condition.

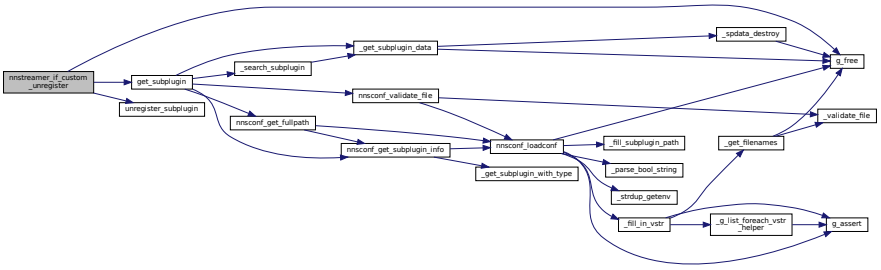
Unregister the custom callback function.

Returns

0 if success. -ERRNO if error.

Definition at line 1049 of file gsttensor_if.c.

Here is the call graph for this function:



9.40.4.29 switch()

```
switch (
            cv-> type )
```

Definition at line 851 of file gsttensor_if.c.

Here is the caller graph for this function:

9.40.5 Variable Documentation

9.40.5.1 cv

```
case tensor_data_s* cv
```

Definition at line 838 of file gsttensor_if.c.

9.40.5.2 data

```
svtc_2 data = tensor_if->sv->data[0]
```

Definition at line 844 of file gsttensor_if.c.

9.40.5.3 result

```
* result
```

Initial value:

```
{
  gboolean ret = FALSE
```

Definition at line 839 of file gsttensor_if.c.

9.40.5.4 sink_factory

```
GstStaticPadTemplate sink_factory  [static]
```

Initial value:

```
= GST_STATIC_PAD_TEMPLATE ("sink",
    GST_PAD_SINK,
    GST_PAD_ALWAYS,
    GST_STATIC_CAPS (CAPS_STRING))
```

The capabilities of the inputs.

Definition at line 109 of file gsttensor_if.c.

9.40.5.5 src_factory

```
GstStaticPadTemplate src_factory  [static]
```

Initial value:

```
= GST_STATIC_PAD_TEMPLATE ("src",
    GST_PAD_SRC,
    GST_PAD_SOMETIMES,
    GST_STATIC_CAPS (CAPS_STRING))
```

The capabilities of the outputs.

Definition at line 117 of file gsttensor_if.c.

9.40.5.6 svtc_1

```
gst_tensor_data_typecast & svtc_1
```

Definition at line 841 of file gsttensor_if.c.

9.40.5.7 svtc_2

```
gst_tensor_data_typecast & svtc_2
```

Definition at line 841 of file gsttensor_if.c.

9.40.5.8 TIFOP_NE

```
case TIFOP_NE
```

t == sv1._##t) ? TRUE : FALSE; break; \

Definition at line 837 of file gsttensor_if.c.

9.40.5.9 TRUE

```
return TRUE
```

Definition at line 897 of file gsttensor_if.c.

9.40.5.10 type

```
svtc_2 type = tensor_if->sv->type
```

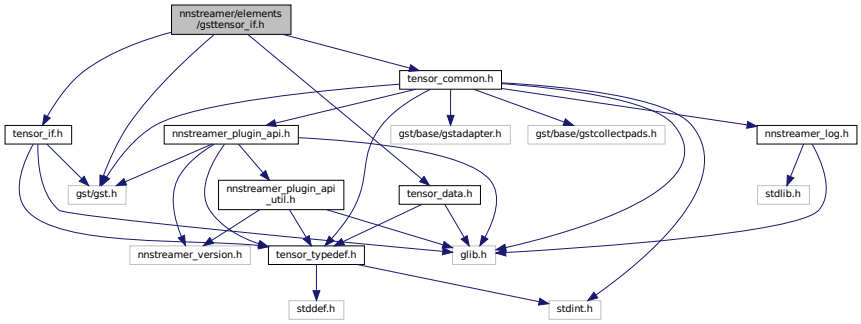
Definition at line 843 of file gsttensor_if.c.

9.41 nnstreamer/elements/gsttensor_if.h File Reference

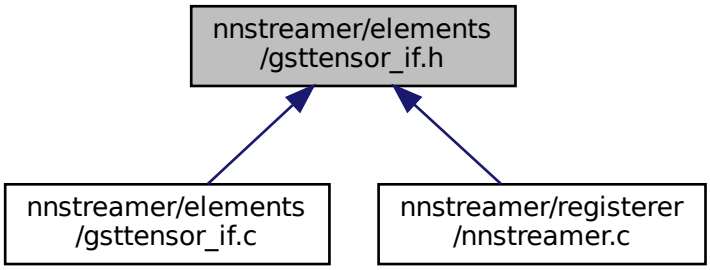
GStreamer plugin to control flow based on tensor values.

```
#include <gst/gst.h>
#include <tensor_common.h>
#include <tensor_data.h>
#include <tensor_if.h>
```

Include dependency graph for gsttensor_if.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct tensor_if_sv_s

  Internal data structure for supplied value.
- struct custom_cb_s
- struct _GstTensorIf

  Tensor If data structure.
- struct _GstTensorIfClass

  GstTensorIfClass inherits GstElementClass.

Macros

- #define GST_TYPE_TENSOR_IF (gst_tensor_if_get_type ())
- #define GST_TENSOR_IF(obj) (G_TYPE_CHECK_INSTANCE_CAST ((obj), GST_TYPE_TENSOR_IF, GstTensorIf))
- #define GST_TENSOR_IF_CLASS(klass) (G_TYPE_CHECK_CLASS_CAST ((klass), GST_TYPE_TENSOR_IF, GstTensorIfClass))
- #define GST_TENSOR_IF_GET_CLASS(obj) (G_TYPE_INSTANCE_GET_CLASS ((obj), GST_TYPE_TENSOR_IF, GstTensorIfClass))
- #define GST_IS_TENSOR_IF(obj) (G_TYPE_CHECK_INSTANCE_TYPE((obj),GST_TYPE_TENSOR_IF))
- #define GST_IS_TENSOR_IF_CLASS(klass) (G_TYPE_CHECK_CLASS_TYPE((klass),GST_TYPE_TENSOR_IF))
- #define GST_TENSOR_IF_CAST(obj) ((GstTensorIf*)(obj))

Typedefs

- typedef struct _GstTensorIf GstTensorIf
- typedef struct _GstTensorIfClass GstTensorIfClass

Enumerations

- enum tensor_if_compared_value {
  TIFCV_A_VALUE = 0, TIFCV_TENSOR_TOTAL_VALUE = 1, TIFCV_ALL_TENSORS_TOTAL_VALUE = 2,
  TIFCV_TENSOR_AVERAGE_VALUE = 3,
  TIFCV_ALL_TENSORS_AVERAGE_VALUE = 4, TIFCV_CUSTOM = 5, TIFCV_END }

  Compared_Value.
- enum tensor_if_operator {
  TIFOP_EQ = 0, TIFOP_NE, TIFOP_GT, TIFOP_GE,
  TIFOP_LT, TIFOP_LE, TIFOP_RANGE_INCLUSIVE, TIFOP_RANGE_EXCLUSIVE,
  TIFOP_NOT_IN_RANGE_INCLUSIVE, TIFOP_NOT_IN_RANGE_EXCLUSIVE, TIFOP_END }

  OPERAND.
- enum tensor_if_behavior {
  TIFB_PASSTHROUGH = 0, TIFB_SKIP, TIFB_FILL_ZERO, TIFB_FILL_VALUES,
  TIFB_FILL_WITH_FILE, TIFB_FILL_WITH_FILE_RPT, TIFB_REPEAT_PREVIOUS_FRAME, TIFB_TENSORPICK,
  TIFB_END }

  Behaviors that may fit in THEN and ELSE.
- enum tensor_if_srcpads { TIFSP_THEN_PAD = 0, TIFSP_ELSE_PAD }

  Which src pad.

Functions

- GType gst_tensor_if_get_type (void)

  Get Type function required for gst elements.

9.41.1 Detailed Description

GStreamer plugin to control flow based on tensor values.

GStreamer/NNStreamer Tensor-IF Copyright (C) 2020 MyungJoo Ham myungjoo.ham@samsung.com

Date

08 April 2020

See also

https://github.com/nnstreamer/nnstreamer

Author

MyungJoo Ham myungjoo.ham@samsung.com

Bug No known bugs except for NYI items

Todo Add "event/signal" to reload FILL_WITH_FILE∗ file??? (TBD)

The output dimension is SAME with input dimension.

9.41.2 Macro Definition Documentation

9.41.2.1 GST_IS_TENSOR_IF

```
#define GST_IS_TENSOR_IF(
              obj ) (G_TYPE_CHECK_INSTANCE_TYPE((obj),GST_TYPE_TENSOR_IF))
```

Definition at line 33 of file gsttensor_if.h.

9.41.2.2 GST_IS_TENSOR_IF_CLASS

```
#define GST_IS_TENSOR_IF_CLASS(
              klass ) (G_TYPE_CHECK_CLASS_TYPE((klass),GST_TYPE_TENSOR_IF))
```

Definition at line 34 of file gsttensor_if.h.

9.41.2.3 GST_TENSOR_IF

```
#define GST_TENSOR_IF(
              obj ) (G_TYPE_CHECK_INSTANCE_CAST ((obj), GST_TYPE_TENSOR_IF, GstTensorIf))
```

Definition at line 30 of file gsttensor_if.h.

9.41.2.4 GST_TENSOR_IF_CAST

```
#define GST_TENSOR_IF_CAST(
              obj ) ((GstTensorIf*)(obj))
```

Definition at line 35 of file gsttensor_if.h.

9.41.2.5 GST_TENSOR_IF_CLASS

```
#define GST_TENSOR_IF_CLASS(
              klass ) (G_TYPE_CHECK_CLASS_CAST ((klass), GST_TYPE_TENSOR_IF, GstTensorIfClass))
```

Definition at line 31 of file gsttensor_if.h.

9.41.2.6 GST_TENSOR_IF_GET_CLASS

```
#define GST_TENSOR_IF_GET_CLASS(
              obj ) (G_TYPE_INSTANCE_GET_CLASS ((obj), GST_TYPE_TENSOR_IF, GstTensorIfClass))
```

Definition at line 32 of file gsttensor_if.h.

9.41.2.7 GST_TYPE_TENSOR_IF

```
#define GST_TYPE_TENSOR_IF (gst_tensor_if_get_type ())
```

Definition at line 29 of file gsttensor_if.h.

9.41.3 Typedef Documentation

9.41.3.1 GstTensorIf

```
typedef struct _GstTensorIf GstTensorIf
```

Definition at line 36 of file gsttensor_if.h.

9.41.3.2 GstTensorIfClass

```
typedef struct _GstTensorIfClass GstTensorIfClass
```

Definition at line 37 of file gsttensor_if.h.

9.41.4 Enumeration Type Documentation

9.41.4.1 tensor_if_behavior

```
enum tensor_if_behavior
```

Behaviors that may fit in THEN and ELSE.

FILL_WITH_FILE, FILL_WITH_FILE_RPT, and REPEAT_PREVIOUS_FRAME caches an output frame and thus, may consume additional memory and incur an additional memcpy.

Enumerator

TIFB_PASSTHROUGH	The input frame becomes the output frame
TIFB_SKIP	Do not generate output frame (frame skip)
TIFB_FILL_ZERO	Fill output frame with zeros
TIFB_FILL_VALUES	Fill output frame with a user given value
TIFB_FILL_WITH_FILE	Fill output frame with a user given file (a raw data of tensor/tensors) If the filesize is smaller, the reset is filled with 0
TIFB_FILL_WITH_FILE_RPT	Fill output frame with a user given file (a raw data of tensor/tensors) If the filesize is smally, the file is repeatedly used
TIFB_REPEAT_PREVIOUS_FRAME	Resend the previous output frame. If this is the first, send ZERO values.
TIFB_TENSORPICK	Choose nth tensor (or tensors) among tensors
TIFB_END	

Definition at line 79 of file gsttensor_if.h.

9.41.4.2 tensor_if_compared_value

```
enum tensor_if_compared_value
```

Compared_Value.

Enumerator

TIFCV_A_VALUE	Decide based on a single scalar value of tensors
TIFCV_TENSOR_TOTAL_VALUE	Decide based on a total (sum) value of a specific tensor
TIFCV_ALL_TENSORS_TOTAL_VALUE	Decide based on a total (sum) value of of tensors or a specific tensor
TIFCV_TENSOR_AVERAGE_VALUE	Decide based on a average value of a specific tensor
TIFCV_ALL_TENSORS_AVERAGE_VALUE	Decide based on a average value of tensors or a specific tensor
TIFCV_CUSTOM	Decide based on a user defined condition
TIFCV_END	

Definition at line 42 of file gsttensor_if.h.

9.41.4.3 tensor_if_operator

`enum tensor_if_operator`

OPERAND.

Enumerator

TIFOP_EQ	==
TIFOP_NE	!=
TIFOP_GT	
TIFOP_GE	>=
TIFOP_LT	<
TIFOP_LE	<=
TIFOP_RANGE_INCLUSIVE	in [min, max]
TIFOP_RANGE_EXCLUSIVE	in (min, max)
TIFOP_NOT_IN_RANGE_INCLUSIVE	not in [min, max]
TIFOP_NOT_IN_RANGE_EXCLUSIVE	not in (min, max)
TIFOP_END	

Definition at line 60 of file gsttensor_if.h.

9.41.4.4 tensor_if_srcpads

`enum tensor_if_srcpads`

Which src pad.

Enumerator

TIFSP_THEN_PAD	
TIFSP_ELSE_PAD	

Definition at line 96 of file gsttensor_if.h.

9.41.5 Function Documentation

9.41.5.1 gst_tensor_if_get_type()

```
GType gst_tensor_if_get_type (
            void  )
```

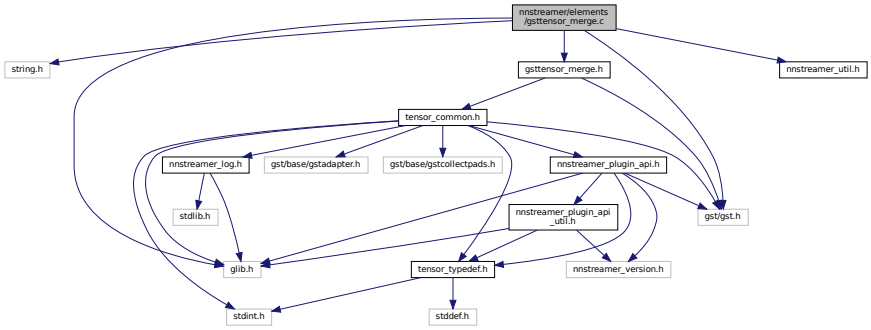
Get Type function required for gst elements.

9.42 nnstreamer/elements/gsttensor_merge.c File Reference

GStreamer plugin to merge tensors (as a filter for other general neural network filters)

```
#include <string.h>
#include <gst/gst.h>
#include <glib.h>
#include <nnstreamer_util.h>
#include "gsttensor_merge.h"
```

Include dependency graph for gsttensor_merge.c:



Macros

- #define GST_CAT_DEFAULT gst_tensor_merge_debug
- #define DBG (!tensor_merge->silent)

  Macro for debug mode.
- #define CAPS_STRING GST_TENSOR_CAP_DEFAULT ";" GST_TENSORS_CAP_WITH_NUM ("1")

  Template caps string.
- #define gst_tensor_merge_parent_class parent_class

Enumerations

- enum {
  PROP_0, PROP_MODE, PROP_OPTION, PROP_SYNC_MODE,
  PROP_SYNC_OPTION, PROP_SILENT }

Functions

- GST_DEBUG_CATEGORY_STATIC (gst_tensor_merge_debug)
- static gboolean gst_tensor_merge_src_event (GstPad *pad, GstObject *parent, GstEvent *event)

  src event vmethod
- static GstPad * gst_tensor_merge_request_new_pad (GstElement *element, GstPadTemplate *templ, const gchar *req_name, const GstCaps *caps)

  making new request pad (gst element vmethod)
- static GstStateChangeReturn gst_tensor_merge_change_state (GstElement *element, GstStateChange transition)

  change state (gst element vmethod)
- static gboolean gst_tensor_merge_sink_event (GstCollectPads *pads, GstCollectData *data, GstEvent *event, GstTensorMerge *tensor_merge)

  sink event vmethod
- static GstFlowReturn gst_tensor_merge_collected (GstCollectPads *pads, GstTensorMerge *tensor_merge)

  Gst Collect Pads Function which is called once collect pads done.
- static void gst_tensor_merge_set_property (GObject *object, guint prop_id, const GValue *value, GParam↩ Spec *pspec)

  Get property (gst element vmethod)
- static void gst_tensor_merge_get_property (GObject *object, guint prop_id, GValue *value, GParamSpec *pspec)

  Get property (gst element vmethod)
- static void gst_tensor_merge_finalize (GObject *object)

  finalize vmethod
- G_DEFINE_TYPE (GstTensorMerge, gst_tensor_merge, GST_TYPE_ELEMENT)
- static void gst_tensor_merge_class_init (GstTensorMergeClass *klass)

  initialize the tensor_merge's class
- static void gst_tensor_merge_init (GstTensorMerge *tensor_merge)

  initialize the new element instantiate pads and add them to element set pad callback functions initialize instance structure
- static tensor_merge_mode gst_tensor_merge_get_mode (const gchar *str)

  Get the corresponding mode from the string value.
- static gboolean gst_tensor_merge_get_merged_config (GstTensorMerge *tensor_merge, const GstTensorsConfig *in_config, GstTensorsConfig *out_config)

  Generate out TensorsConfig with in TensorsConfig.
- static gboolean gst_tensor_merge_collect_buffer (GstTensorMerge *tensor_merge, GstBuffer *tensors_buf, gboolean *is_eos)

  Looping to generete output buffer for srcpad.
- static GstFlowReturn gst_tensor_merge_generate_mem (GstTensorMerge *tensor_merge, GstBuffer *tensors_buf, GstBuffer *tensor_buf)

  Generate Output GstMemory.
- static gboolean gst_tensor_merge_set_src_caps (GstTensorMerge *tensor_merge)

  Set src pad caps if src pad is not negotiated.
- static void gst_tensor_merge_send_segment_event (GstTensorMerge *tensor_merge, GstClockTime pts, GstClockTime dts)

  Send segment event if necessary.
- static void gst_tensor_merge_ready_to_paused (GstTensorMerge *tensor_merge)

  Ready --> Pasuse State Change.
- static gboolean gst_tensor_merge_set_option_data (GstTensorMerge *tensor_merge)

  Setup internal data (data_* in GstTensor_Merge)

Variables

- static const gchar * gst_tensor_merge_mode_string [ ]
- static const gchar * gst_tensor_merge_linear_string [ ]
- static GstStaticPadTemplate src_templ

  the capabilities of the inputs and outputs. describe the real formats here.
- static GstStaticPadTemplate sink_templ

9.42.1 Detailed Description

GStreamer plugin to merge tensors (as a filter for other general neural network filters)

GStreamer Copyright (C) 2005 Thomas Vander Stichele thomas@apestaart.org Copyright (C) 2005 Ronald S. Bultje rbultje@ronald.bitfreak.net Copyright (C) 2018 Jijoong Moon jijoong.moon@samsung.com

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Library General Public License as published by the Free Software Foundation; version 2.1 of the License.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Library General Public License for more details.

Date

03 July 2018

See also

https://github.com/nnstreamer/nnstreamer

Author

Jijoong Moon jijoong.moon@samsung.com

Bug No known bugs except for NYI items

9.42.2 Macro Definition Documentation

9.42.2.1 CAPS_STRING

```
#define CAPS_STRING GST_TENSOR_CAP_DEFAULT ";" GST_TENSORS_CAP_WITH_NUM ("1")
```

Template caps string.

Definition at line 102 of file gsttensor_merge.c.

9.42.2.2 DBG

```
#define DBG (!tensor_merge->silent)
```

Macro for debug mode.

Definition at line 73 of file gsttensor_merge.c.

9.42.2.3 GST_CAT_DEFAULT

```
#define GST_CAT_DEFAULT gst_tensor_merge_debug
```

Definition at line 67 of file gsttensor_merge.c.

9.42.2.4 gst_tensor_merge_parent_class

```
#define gst_tensor_merge_parent_class parent_class
```

Definition at line 135 of file gsttensor_merge.c.

9.42.3 Enumeration Type Documentation

9.42.3.1 anonymous enum

```
anonymous enum
```

Enumerator

PROP_0	
PROP_MODE	
PROP_OPTION	
PROP_SYNC_MODE	
PROP_SYNC_OPTION	
PROP_SILENT	

Definition at line 76 of file gsttensor_merge.c.

9.42.4 Function Documentation

9.42.4.1 G_DEFINE_TYPE()

```
G_DEFINE_TYPE (
            GstTensorMerge ,
            gst_tensor_merge ,
            GST_TYPE_ELEMENT  )
```

9.42.4.2 GST_DEBUG_CATEGORY_STATIC()

```
GST_DEBUG_CATEGORY_STATIC (
            gst_tensor_merge_debug  )
```

SECTION:element-tensormerge

A Merger that merge tensor stream to tensor stream for NN frameworks. The output is always in the format of other/tensor

<refsect2> <title>Example launch line</title> |[gst-launch -v -m tensor_merge name=merge ! fakesink filesrc location=b.png ! pngdec ! videoscale ! imagefreeze ! videoconvert ! video/x-raw,format=R↩ GB,width=100,height=100,framerate=0/1 ! tensor_converter ! merge.sink_0 filesrc location=b.png ! pngdec ! videoscale ! imagefreeze ! videoconvert ! video/x-raw,format=RGB,width=100,height=100,framerate=0/1 ! tensor_converter ! merge.sink_1 filesrc location=b.png ! pngdec ! videoscale ! imagefreeze ! videoconvert ! video/x-raw,format=RGB,width=100,height=100,framerate=0/1 ! tensor_converter ! merge.sink_2]|

|[gst-launch -v -m tensor_merge name=merge ! filesink location=merge.log multifilesrc location="testsequence↩ _%1d.png" index=0 caps="image/png, framerate=(fraction)30/1" ! pngdec ! tensor_converter ! merge.sink_↩ 0 multifilesrc location="testsequence_%1d.png" index=0 caps="image/png, framerate=(fraction)30/1" ! pngdec ! tensor_converter ! merge.sink_1 multifilesrc location="testsequence_%1d.png" index=0 caps="image/png, framerate=(fraction)30/1" ! pngdec ! tensor_converter ! merge.sink_2

</refsect2>

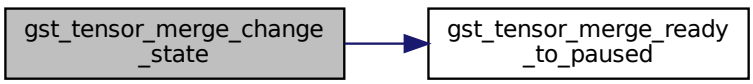
9.42.4.3 gst_tensor_merge_change_state()

```
static GstStateChangeReturn gst_tensor_merge_change_state (
            GstElement * element,
            GstStateChange transition )  [static]
```

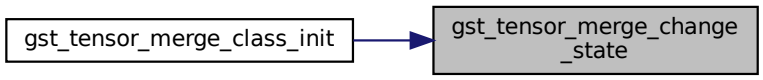
change state (gst element vmethod)

Definition at line 776 of file gsttensor_merge.c.

Here is the call graph for this function:



Here is the caller graph for this function:



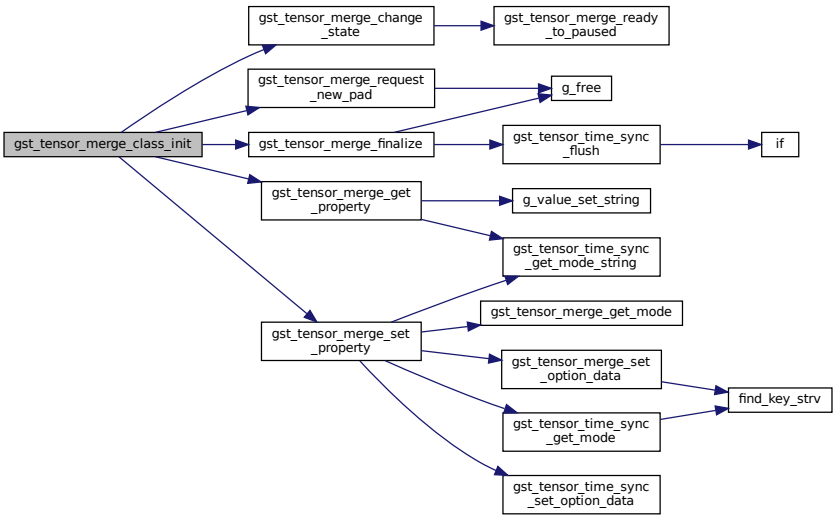
9.42.4.4 gst_tensor_merge_class_init()

```
static void gst_tensor_merge_class_init (
            GstTensorMergeClass * klass )  [static]
```

initialize the tensor_merge's class

Definition at line 142 of file gsttensor_merge.c.

Here is the call graph for this function:



9.42.4.5 gst_tensor_merge_collect_buffer()

```
static gboolean gst_tensor_merge_collect_buffer (
            GstTensorMerge * tensor_merge,
            GstBuffer * tensors_buf,
            gboolean * is_eos )  [static]
```

Looping to generete output buffer for srcpad.

Parameters

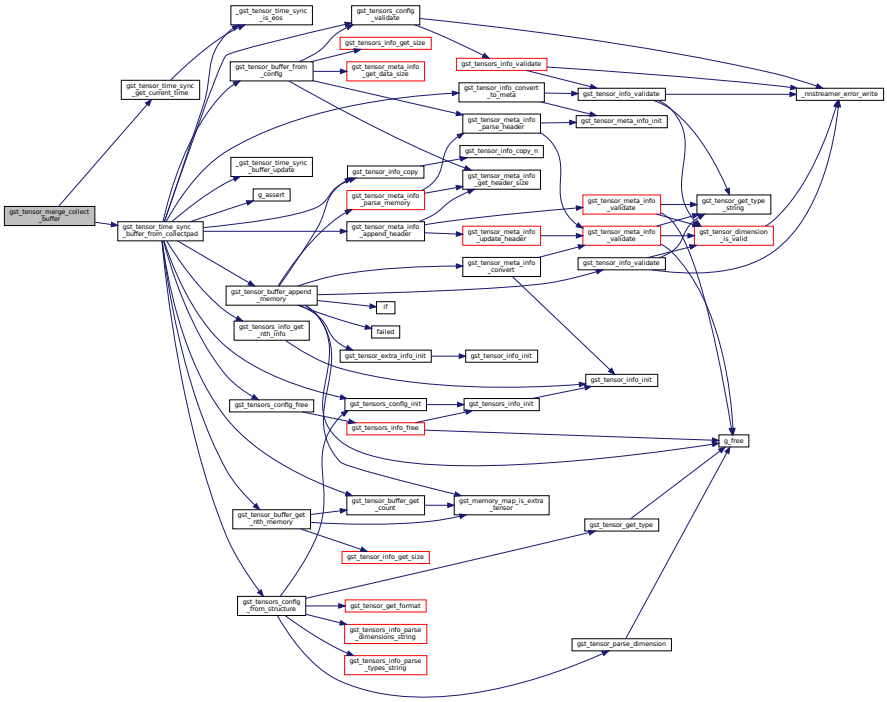
tensor_merge	tensor merger
tensor_buf	output buffer for srcpad
is_eos	boolean EOS (End of Stream)

Returns

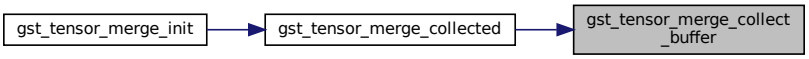
TRUE to push buffer to src pad

Definition at line 448 of file gsttensor_merge.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.42.4.6 gst_tensor_merge_collected()

```
static GstFlowReturn gst_tensor_merge_collected (
            GstCollectPads * pads,
            GstTensorMerge * tensor_merge )  [static]
```

Gst Collect Pads Function which is called once collect pads done.

Parameters

pads	GstCollectPads
tensor_merge	Merger

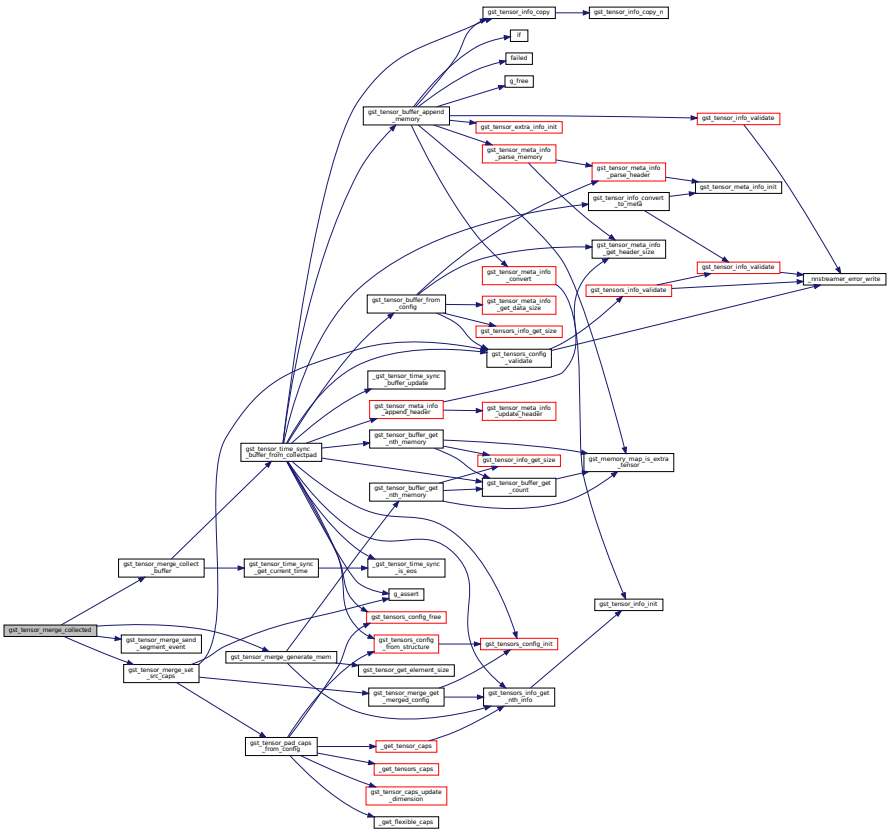
Returns

GstFlowReturn

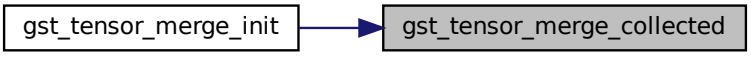
Cannot use gst-pad util to get stream ID (multiple sink pads). Create stream ID using first sink pad.

Definition at line 693 of file gsttensor_merge.c.

Here is the call graph for this function:



Here is the caller graph for this function:



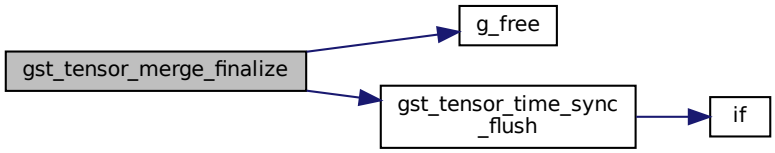
9.42.4.7 gst_tensor_merge_finalize()

```
static void gst_tensor_merge_finalize (
            GObject * object )  [static]
```

finalize vmethod

Definition at line 258 of file gsttensor_merge.c.

Here is the call graph for this function:



Here is the caller graph for this function:

9.42.4.8 gst_tensor_merge_generate_mem()

```
static GstFlowReturn gst_tensor_merge_generate_mem (
            GstTensorMerge * tensor_merge,
            GstBuffer * tensors_buf,
            GstBuffer * tensor_buf )  [static]
```

Generate Output GstMemory.

Parameters

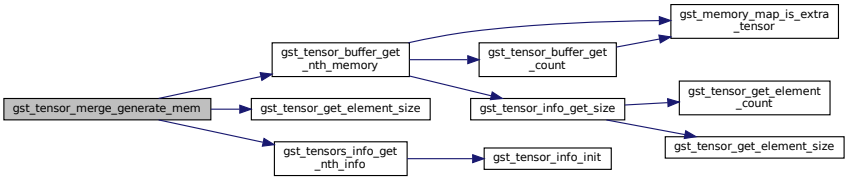
tensor_merge	tensor merger
tensors_buf	collected tensors buffer
tensor_buf	output tensor buffer

Returns

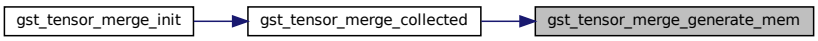
boolean

Definition at line 475 of file gsttensor_merge.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.42.4.9 gst_tensor_merge_get_merged_config()

```
static gboolean gst_tensor_merge_get_merged_config (
            GstTensorMerge * tensor_merge,
            const GstTensorsConfig * in_config,
            GstTensorsConfig * out_config )  [static]
```

Generate out TensorsConfig with in TensorsConfig.

Parameters

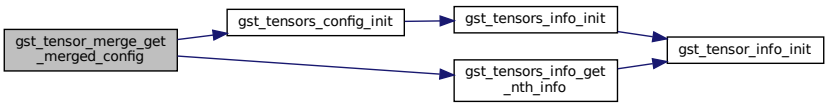
tensor_merge	tensor merger
in_config	in tensors config data (multi tensors)
out_config	out tensors config data (single tensor)

Returns

true / false

Definition at line 379 of file gsttensor_merge.c.

Here is the call graph for this function:



Here is the caller graph for this function:

9.42.4.10 gst_tensor_merge_get_mode()

```
static tensor_merge_mode gst_tensor_merge_get_mode (
            const gchar * str )  [static]
```

Get the corresponding mode from the string value.

Parameters

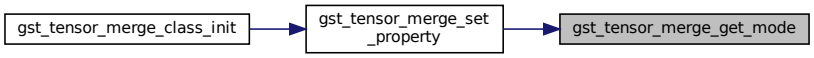
in	*str*	The string value for the mode

Returns

corresponding mode for the string. GTT_END for errors

Definition at line 244 of file gsttensor_merge.c.

Here is the caller graph for this function:



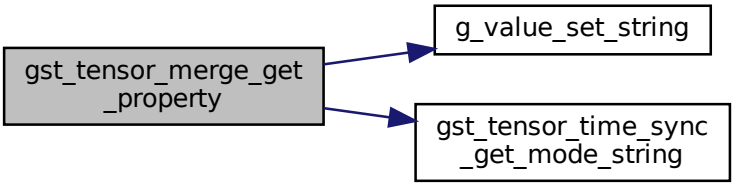
9.42.4.11 gst_tensor_merge_get_property()

```
static void gst_tensor_merge_get_property (
             GObject * object,
             guint prop_id,
             GValue * value,
             GParamSpec * pspec )  [static]
```

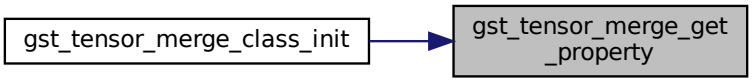
Get property (gst element vmethod)

Definition at line 894 of file gsttensor_merge.c.

Here is the call graph for this function:



Here is the caller graph for this function:



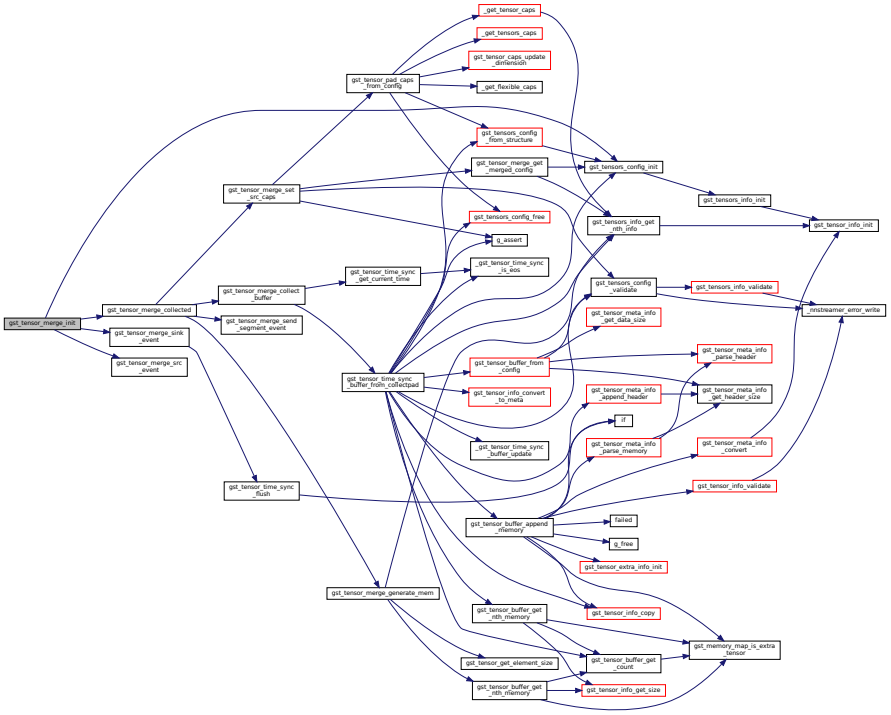
9.42.4.12 gst_tensor_merge_init()

```
static void gst_tensor_merge_init (
             GstTensorMerge * tensor_merge )  [static]
```

initialize the new element instantiate pads and add them to element set pad callback functions initialize instance structure

Definition at line 209 of file gsttensor_merge.c.

Here is the call graph for this function:



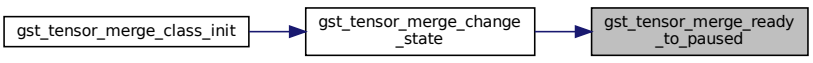
9.42.4.13 gst_tensor_merge_ready_to_paused()

```
static void gst_tensor_merge_ready_to_paused (
            GstTensorMerge * tensor_merge )  [static]
```

Ready --> Pasuse State Change.

Definition at line 764 of file gsttensor_merge.c.

Here is the caller graph for this function:



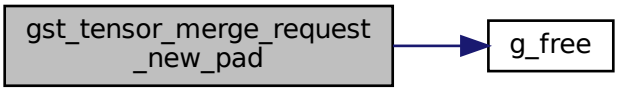
9.42.4.14 gst_tensor_merge_request_new_pad()

```
static GstPad * gst_tensor_merge_request_new_pad (
            GstElement * element,
            GstPadTemplate * templ,
            const gchar * name,
            const GstCaps * caps )  [static]
```

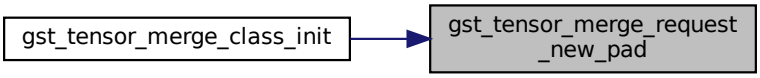
making new request pad (gst element vmethod)

Definition at line 287 of file gsttensor_merge.c.

Here is the call graph for this function:



Here is the caller graph for this function:



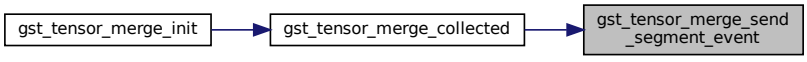
9.42.4.15 gst_tensor_merge_send_segment_event()

```
static void gst_tensor_merge_send_segment_event (
            GstTensorMerge * tensor_merge,
            GstClockTime pts,
            GstClockTime dts )  [static]
```

Send segment event if necessary.

Definition at line 666 of file gsttensor_merge.c.

Here is the caller graph for this function:



9.42.4.16 gst_tensor_merge_set_option_data()

```
static gboolean gst_tensor_merge_set_option_data (
            GstTensorMerge * tensor_merge )  [static]
```

Setup internal data (data_∗ in GstTensor_Merge)

Parameters

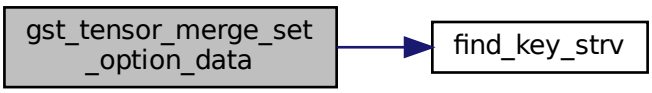
[in/out]	filter "this" pointer. mode & option MUST BE set already.

Return values

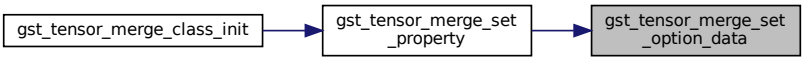
TRUE	if ok or not configured, yet.
FALSE	if given input is configured invalid.

Definition at line 812 of file gsttensor_merge.c.

Here is the call graph for this function:



Here is the caller graph for this function:



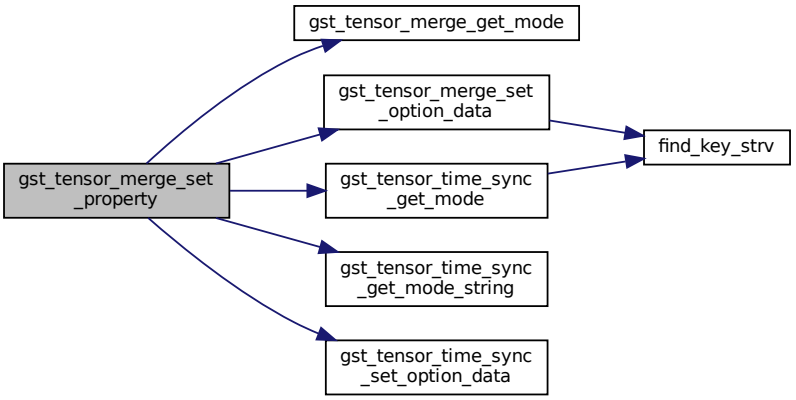
9.42.4.17 gst_tensor_merge_set_property()

```
static void gst_tensor_merge_set_property (
            GObject * object,
            guint prop_id,
            const GValue * value,
            GParamSpec * pspec )  [static]
```

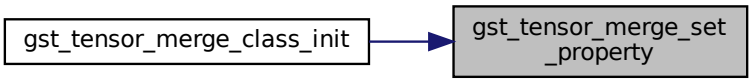
Get property (gst element vmethod)

Definition at line 841 of file gsttensor_merge.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.42.4.18 gst_tensor_merge_set_src_caps()

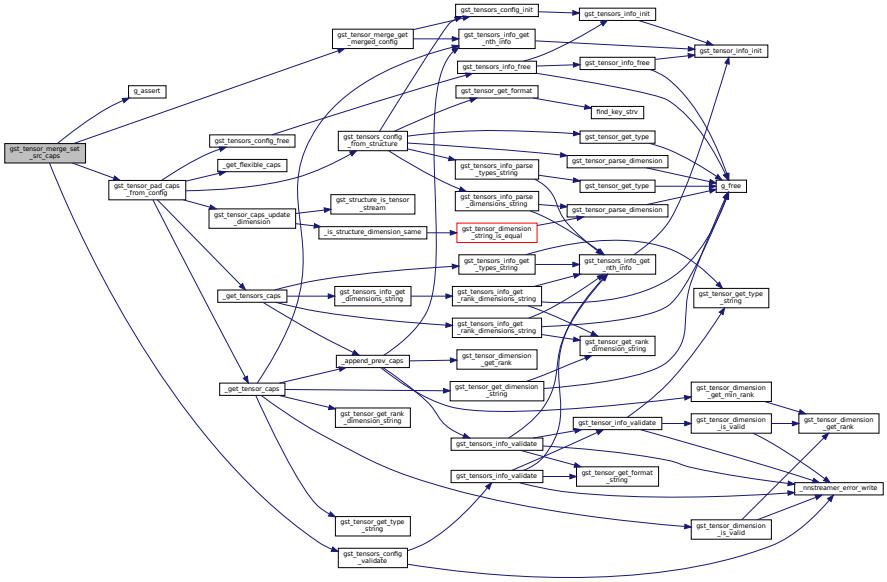
```
static gboolean gst_tensor_merge_set_src_caps (
            GstTensorMerge * tensor_merge )  [static]
```

Set src pad caps if src pad is not negotiated.

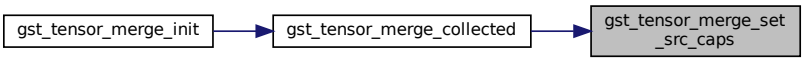
Internal Logic Error?

Definition at line 632 of file gsttensor_merge.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.42.4.19 gst_tensor_merge_sink_event()

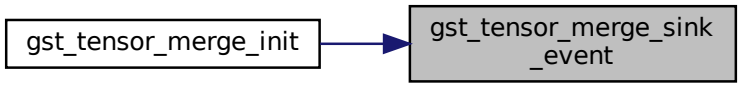
```
static gboolean gst_tensor_merge_sink_event (
            GstCollectPads * pads,
            GstCollectData * data,
            GstEvent * event,
            GstTensorMerge * tensor_merge )  [static]
```

sink event vmethod

Definition at line 353 of file gsttensor_merge.c.

Here is the call graph for this function:

Here is the caller graph for this function:



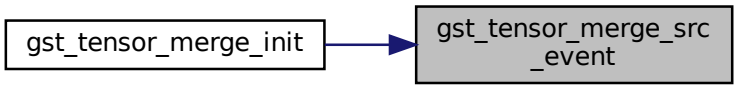
9.42.4.20 gst_tensor_merge_src_event()

```
static gboolean gst_tensor_merge_src_event (
             GstPad * pad,
             GstObject * parent,
             GstEvent * event )  [static]
```

src event vmethod

Definition at line 334 of file gsttensor_merge.c.

Here is the caller graph for this function:



9.42.5 Variable Documentation

9.42.5.1 gst_tensor_merge_linear_string

```
const gchar* gst_tensor_merge_linear_string[]  [static]
```

Initial value:

```
= {
  [LINEAR_FIRST] = "0",
  [LINEAR_SECOND] = "1",
  [LINEAR_THIRD] = "2",
  [LINEAR_FOURTH] = "3",
  [LINEAR_END] = NULL,
}
```

Definition at line 91 of file gsttensor_merge.c.

9.42.5.2 gst_tensor_merge_mode_string

```
const gchar* gst_tensor_merge_mode_string[]  [static]
```

Initial value:

```
= {
  [GTT_LINEAR] = "linear",
  [GTT_END] = "error",
}
```

Definition at line 86 of file gsttensor_merge.c.

9.42.5.3 sink_templ

```
GstStaticPadTemplate sink_templ  [static]
```

Initial value:

```
= GST_STATIC_PAD_TEMPLATE ("sink_%u",
    GST_PAD_SINK,
    GST_PAD_REQUEST,
    GST_STATIC_CAPS (CAPS_STRING))
```

Definition at line 113 of file gsttensor_merge.c.

9.42.5.4 src_templ

```
GstStaticPadTemplate src_templ  [static]
```

Initial value:

```
= GST_STATIC_PAD_TEMPLATE ("src",
    GST_PAD_SRC,
    GST_PAD_ALWAYS,
    GST_STATIC_CAPS (CAPS_STRING))
```

the capabilities of the inputs and outputs. describe the real formats here.

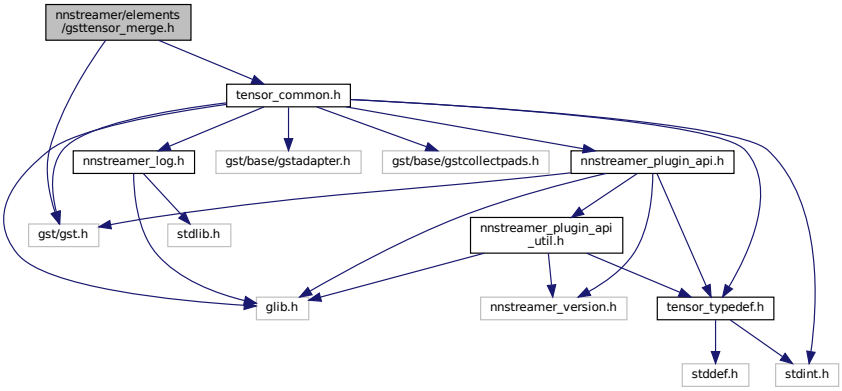
Definition at line 108 of file gsttensor_merge.c.

9.43 nnstreamer/elements/gsttensor_merge.h File Reference

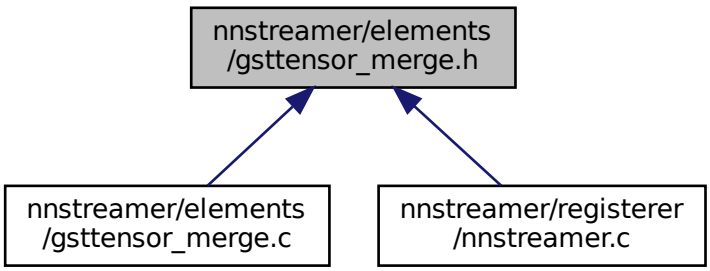
GStreamer plugin to merge tensors (as a filter for other general neural network filters)

```
#include <gst/gst.h>
#include <tensor_common.h>
```

Include dependency graph for gsttensor_merge.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct _tensor_merge_linear

  Internal data structure for linear mode.
- struct _GstTensorMerge

  Tensor Merge data structure.
- struct _GstTensorMergeClass

  GstTensorMergeClass inherits GstElementClass.

Macros

- #define GST_TYPE_TENSOR_MERGE (gst_tensor_merge_get_type ())
- #define GST_TENSOR_MERGE(obj) (G_TYPE_CHECK_INSTANCE_CAST ((obj), GST_TYPE_TENSOR_MERGE, GstTensorMerge))
- #define GST_TENSOR_MERGE_CLASS(klass) (G_TYPE_CHECK_CLASS_CAST ((klass), GST_TYPE_TENSOR_MERGE, GstTensorMergeClass))
- #define GST_TENSOR_MERGE_GET_CLASS(obj) (G_TYPE_INSTANCE_GET_CLASS ((obj), GST_TYPE_TENSOR_MERG GstTensorMergeClass))
- #define GST_IS_TENSOR_MERGE(obj) (G_TYPE_CHECK_INSTANCE_TYPE((obj),GST_TYPE_TENSOR_MERGE))
- #define GST_IS_TENSOR_MERGE_CLASS(klass) (G_TYPE_CHECK_CLASS_TYPE((klass),GST_TYPE_TENSOR_MERGE
- #define GST_TENSOR_MERGE_CAST(obj) ((GstTensorMerge*)(obj))

Typedefs

- typedef struct _GstTensorMerge GstTensorMerge
- typedef struct _GstTensorMergeClass GstTensorMergeClass
- typedef struct _tensor_merge_linear tensor_merge_linear

  Internal data structure for linear mode.

Enumerations

- enum tensor_merge_mode { GTT_LINEAR = 0, GTT_END }
- enum tensor_merge_linear_mode {
  LINEAR_FIRST = 0, LINEAR_SECOND = 1, LINEAR_THIRD = 2, LINEAR_FOURTH = 3,
  LINEAR_END }

Functions

- GType gst_tensor_merge_get_type (void)

  Get Type function required for gst elements.

9.43.1 Detailed Description

GStreamer plugin to merge tensors (as a filter for other general neural network filters)

GStreamer Copyright (C) 2005 Thomas Vander Stichele thomas@apestaart.org Copyright (C) 2005 Ronald S. Bultje rbultje@ronald.bitfreak.net Copyright (C) 2018 Jijoong Moon jijoong.moon@samsung.com

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Library General Public License as published by the Free Software Foundation; version 2.1 of the License.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Library General Public License for more details.

Date

03 July 2018

See also

https://github.com/nnstreamer/nnstreamer

Author

Jijoong Moon jijoong.moon@samsung.com

Bug No known bugs except for NYI items

9.43.2 Macro Definition Documentation

9.43.2.1 GST_IS_TENSOR_MERGE

```
#define GST_IS_TENSOR_MERGE(
              obj ) (G_TYPE_CHECK_INSTANCE_TYPE((obj),GST_TYPE_TENSOR_MERGE))
```

Definition at line 39 of file gsttensor_merge.h.

9.43.2.2 GST_IS_TENSOR_MERGE_CLASS

```
#define GST_IS_TENSOR_MERGE_CLASS(
              klass ) (G_TYPE_CHECK_CLASS_TYPE((klass),GST_TYPE_TENSOR_MERGE))
```

Definition at line 40 of file gsttensor_merge.h.

9.43.2.3 GST_TENSOR_MERGE

```
#define GST_TENSOR_MERGE(
              obj ) (G_TYPE_CHECK_INSTANCE_CAST ((obj), GST_TYPE_TENSOR_MERGE, GstTensorMerge))
```

Definition at line 36 of file gsttensor_merge.h.

9.43.2.4 GST_TENSOR_MERGE_CAST

```
#define GST_TENSOR_MERGE_CAST(
              obj ) ((GstTensorMerge*)(obj))
```

Definition at line 41 of file gsttensor_merge.h.

9.43.2.5 GST_TENSOR_MERGE_CLASS

```
#define GST_TENSOR_MERGE_CLASS(
              klass ) (G_TYPE_CHECK_CLASS_CAST ((klass), GST_TYPE_TENSOR_MERGE, GstTensorMergeClass))
```

Definition at line 37 of file gsttensor_merge.h.

9.43.2.6 GST_TENSOR_MERGE_GET_CLASS

```
#define GST_TENSOR_MERGE_GET_CLASS(
              obj ) (G_TYPE_INSTANCE_GET_CLASS ((obj), GST_TYPE_TENSOR_MERGE, GstTensorMergeClass))
```

Definition at line 38 of file gsttensor_merge.h.

9.43.2.7 GST_TYPE_TENSOR_MERGE

```
#define GST_TYPE_TENSOR_MERGE (gst_tensor_merge_get_type ())
```

Definition at line 35 of file gsttensor_merge.h.

9.43.3 Typedef Documentation

9.43.3.1 GstTensorMerge

```
typedef struct _GstTensorMerge GstTensorMerge
```

Definition at line 42 of file gsttensor_merge.h.

9.43.3.2 GstTensorMergeClass

```
typedef struct _GstTensorMergeClass GstTensorMergeClass
```

Definition at line 43 of file gsttensor_merge.h.

9.43.3.3 tensor_merge_linear

```
typedef struct _tensor_merge_linear tensor_merge_linear
```

Internal data structure for linear mode.

9.43.4 Enumeration Type Documentation

9.43.4.1 tensor_merge_linear_mode

```
enum tensor_merge_linear_mode
```

Enumerator

LINEAR_FIRST	
LINEAR_SECOND	
LINEAR_THIRD	
LINEAR_FOURTH	
LINEAR_END	

Definition at line 51 of file gsttensor_merge.h.

9.43.4.2 tensor_merge_mode

```
enum tensor_merge_mode
```

Enumerator

GTT_LINEAR	
GTT_END	

Definition at line 45 of file gsttensor_merge.h.

9.43.5 Function Documentation

9.43.5.1 gst_tensor_merge_get_type()

```
GType gst_tensor_merge_get_type (
            void  )
```

Get Type function required for gst elements.

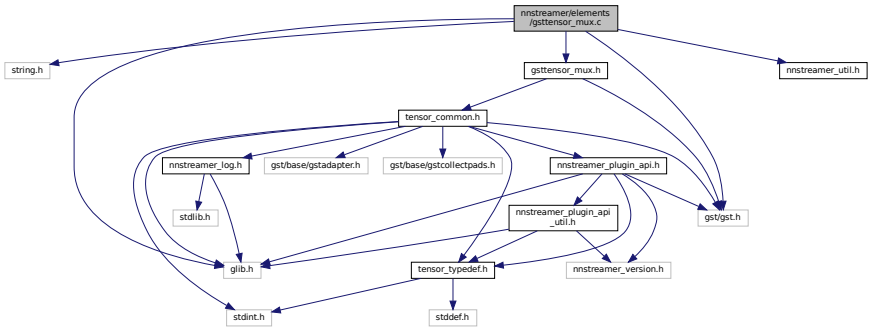
9.44 nnstreamer/elements/gsttensor_mux.c File Reference

GStreamer plugin to mux tensors (as a filter for other general neural network filters)

```
#include <string.h>
#include <gst/gst.h>
#include <glib.h>
#include <nnstreamer_util.h>
```

```
#include "gsttensor_mux.h"
```

Include dependency graph for gsttensor_mux.c:



Macros

- #define GST_CAT_DEFAULT gst_tensor_mux_debug
- #define DBG (!tensor_mux->silent)

  Macro for debug mode.
- #define CAPS_STRING_SINK GST_TENSOR_CAP_DEFAULT ";" GST_TENSORS_CAP_MAKE ("{ static, flexible }")

  Default caps string for sink pad.
- #define CAPS_STRING_SRC GST_TENSORS_CAP_MAKE ("{ static, flexible }")

  Default caps string for src pad.
- #define gst_tensor_mux_parent_class parent_class

Enumerations

- enum { PROP_0, PROP_SILENT, PROP_SYNC_MODE, PROP_SYNC_OPTION }

Functions

- GST_DEBUG_CATEGORY_STATIC (gst_tensor_mux_debug)
- static gboolean gst_tensor_mux_src_event (GstPad *pad, GstObject *parent, GstEvent *event)

  src event vmethod
- static GstPad * gst_tensor_mux_request_new_pad (GstElement *element, GstPadTemplate *templ, const gchar *req_name, const GstCaps *caps)

  making new request pad (gst element vmethod)
- static GstStateChangeReturn gst_tensor_mux_change_state (GstElement *element, GstStateChange transition)

  change state (gst element vmethod)
- static gboolean gst_tensor_mux_sink_event (GstCollectPads *pads, GstCollectData *data, GstEvent *event, GstTensorMux *tensor_mux)

  sink event vmethod
- static GstFlowReturn gst_tensor_mux_collected (GstCollectPads *pads, GstTensorMux *tensor_mux)

  Gst Collect Pads Function which is called once collect pads done.
- static GstFlowReturn gst_tensor_mux_do_clip (GstCollectPads *pads, GstCollectData *data, GstBuffer *buffer, GstBuffer **out, GstTensorMux *tensor_mux)

Gst Clip Pads Function which is called right after a buffer is received for each pad.

- static void gst_tensor_mux_set_property (GObject *object, guint prop_id, const GValue *value, GParamSpec *pspec)

  Get property (gst element vmethod)
- static void gst_tensor_mux_get_property (GObject *object, guint prop_id, GValue *value, GParamSpec *pspec)

  Get property (gst element vmethod)
- static void gst_tensor_mux_finalize (GObject *object)

  finalize vmethod
- G_DEFINE_TYPE (GstTensorMux, gst_tensor_mux, GST_TYPE_ELEMENT)
- static void gst_tensor_mux_class_init (GstTensorMuxClass *klass)

  initialize the tensor_mux's class
- static void gst_tensor_mux_init (GstTensorMux *tensor_mux)

  initialize the new element instantiate pads and add them to element set pad callback functions initialize instance structure
- static void gst_tensor_mux_set_waiting (GstTensorMux *tensor_mux, gboolean waiting)

  set pads waiting property
- static gboolean gst_tensor_mux_collect_buffer (GstTensorMux *tensor_mux, GstBuffer *tensors_buf, gboolean *is_eos)

  Looping to generete output buffer for srcpad.
- static gboolean gst_tensor_mux_set_src_caps (GstTensorMux *tensor_mux)

  Set src pad caps if src pad is not negotiated.
- static void gst_tensor_mux_send_segment_event (GstTensorMux *tensor_mux, GstClockTime pts, GstClockTime dts)

  Create a new segment event if necessary.
- static GstBuffer * gst_tensor_mux_chain_flex_tensor (GstTensorMux *tensor_mux, GstBuffer *buf)

  Process flex tensor.
- static void gst_tensor_mux_ready_to_paused (GstTensorMux *tensor_mux)

  Ready --> Pasuse State Change.

Variables

- static GstStaticPadTemplate src_templ

  the capabilities of the inputs and outputs. describe the real formats here.
- static GstStaticPadTemplate sink_templ

9.44.1 Detailed Description

GStreamer plugin to mux tensors (as a filter for other general neural network filters)

GStreamer Copyright (C) 2005 Thomas Vander Stichele thomas@apestaart.org Copyright (C) 2005 Ronald S. Bultje rbultje@ronald.bitfreak.net Copyright (C) 2018 Jijoong Moon jijoong.moon@samsung.com

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Library General Public License as published by the Free Software Foundation; version 2.1 of the License.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Library General Public License for more details.

Date

03 July 2018

See also

https://github.com/nnstreamer/nnstreamer

Author

Jijoong Moon jijoong.moon@samsung.com

Bug No known bugs except for NYI items

9.44.2 Macro Definition Documentation

9.44.2.1 CAPS_STRING_SINK

```
#define CAPS_STRING_SINK GST_TENSOR_CAP_DEFAULT ";" GST_TENSORS_CAP_MAKE ("{ static, flexible
}")
```

Default caps string for sink pad.

Definition at line 89 of file gsttensor_mux.c.

9.44.2.2 CAPS_STRING_SRC

```
#define CAPS_STRING_SRC GST_TENSORS_CAP_MAKE ("{ static, flexible }")
```

Default caps string for src pad.

Definition at line 94 of file gsttensor_mux.c.

9.44.2.3 DBG

```
#define DBG (!tensor_mux->silent)
```

Macro for debug mode.

Definition at line 75 of file gsttensor_mux.c.

9.44.2.4 GST_CAT_DEFAULT

```
#define GST_CAT_DEFAULT gst_tensor_mux_debug
```

Definition at line 69 of file gsttensor_mux.c.

9.44.2.5 gst_tensor_mux_parent_class

```
#define gst_tensor_mux_parent_class parent_class
```

Definition at line 132 of file gsttensor_mux.c.

9.44.3 Enumeration Type Documentation

9.44.3.1 anonymous enum

```
anonymous enum
```

Enumerator

PROP_0	
PROP_SILENT	
PROP_SYNC_MODE	
PROP_SYNC_OPTION	

Definition at line 78 of file gsttensor_mux.c.

9.44.4 Function Documentation

9.44.4.1 G_DEFINE_TYPE()

```
G_DEFINE_TYPE (
            GstTensorMux ,
            gst_tensor_mux ,
            GST_TYPE_ELEMENT  )
```

9.44.4.2 GST_DEBUG_CATEGORY_STATIC()

```
GST_DEBUG_CATEGORY_STATIC (
            gst_tensor_mux_debug  )
```

SECTION:element-tensormux

A Muxer that merge tensor stream to tensors stream for NN frameworks. The output is always in the format of other/tensors

<refsect2> <title>Example launch line</title> |[gst-launch -v -m \ filesrc location=b.png ! pngdec ! videoscale ! imagefreeze ! videoconvert ! video/x-raw,format=RGB,width=100,height=100,framerate=0/1 ! tensor_converter ! mux.sink_0 \ filesrc location=b.png ! pngdec ! videoscale ! imagefreeze ! videoconvert ! video/x-raw,format=RGB,width=100,height=100,framerate=0/1 ! tensor_converter ! mux.sink_1 \ filesrc location=b.png ! pngdec ! videoscale ! imagefreeze ! videoconvert ! video/x-raw,format=RGB,width=100,height=100,framerate=0/1 ! tensor_converter ! mux.sink_2 \ tensor_mux name=mux ! fakesink]|

|[gst-launch -v -m \ multifilesrc location="testsequence_%1d.png" index=0 caps="image/png, framerate=(fraction)30/1" ! pngdec ! tensor_converter ! mux.sink_0 \ multifilesrc location="testsequence_%1d.png" index=0 caps="image/png, framerate=(fraction)30/1" ! pngdec ! tensor_converter ! mux.sink_1 \ multifilesrc location="testsequence_%1d.png" index=0 caps="image/png, framerate=(fraction)30/1" ! pngdec ! tensor_converter ! mux.sink_2 \ tensor_mux name=mux ! filesink location=mux.log]| </refsect2>

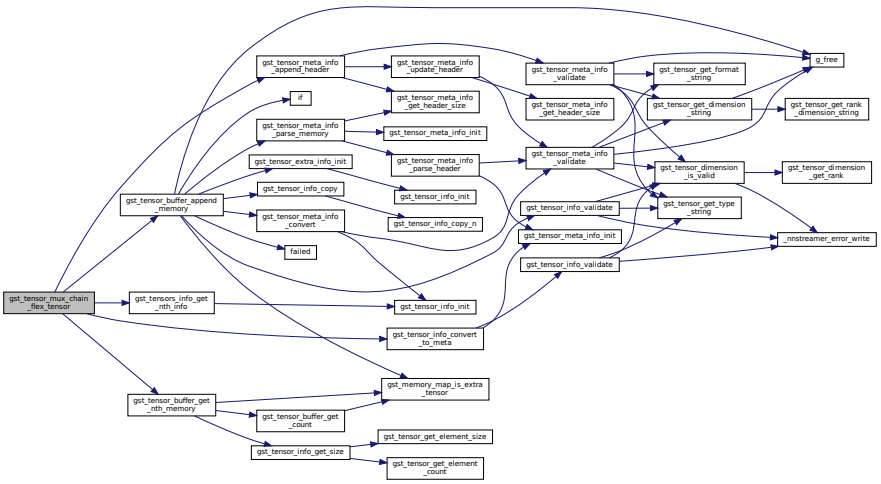
9.44.4.3 gst_tensor_mux_chain_flex_tensor()

```
static GstBuffer* gst_tensor_mux_chain_flex_tensor (
            GstTensorMux * tensor_mux,
            GstBuffer * buf )  [static]
```

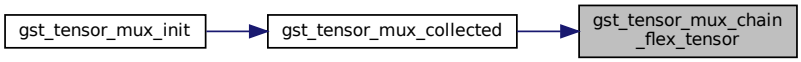
Process flex tensor.

Definition at line 446 of file gsttensor_mux.c.

Here is the call graph for this function:



Here is the caller graph for this function:



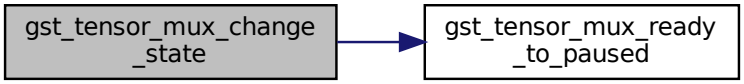
9.44.4.4 gst_tensor_mux_change_state()

```
static GstStateChangeReturn gst_tensor_mux_change_state (
            GstElement * element,
            GstStateChange transition )  [static]
```

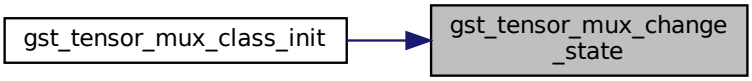
change state (gst element vmethod)

Definition at line 584 of file gsttensor_mux.c.

Here is the call graph for this function:



Here is the caller graph for this function:



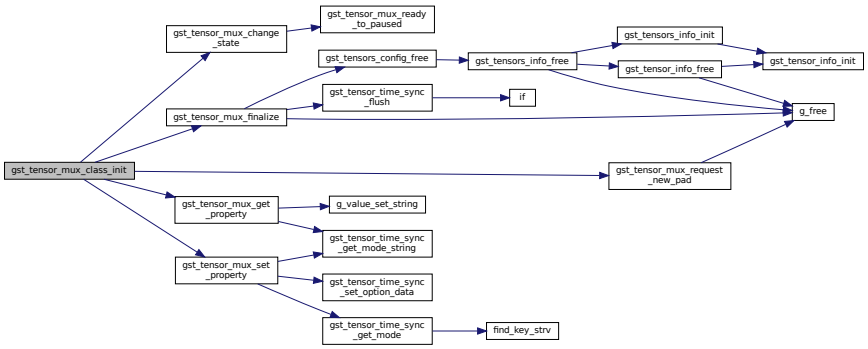
9.44.4.5 gst_tensor_mux_class_init()

```
static void gst_tensor_mux_class_init (
            GstTensorMuxClass * klass )  [static]
```

initialize the tensor_mux's class

Definition at line 139 of file gsttensor_mux.c.

Here is the call graph for this function:



9.44.4.6 gst_tensor_mux_collect_buffer()

```
static gboolean gst_tensor_mux_collect_buffer (
            GstTensorMux * tensor_mux,
            GstBuffer * tensors_buf,
            gboolean * is_eos )  [static]
```

Looping to generete output buffer for srcpad.

Parameters

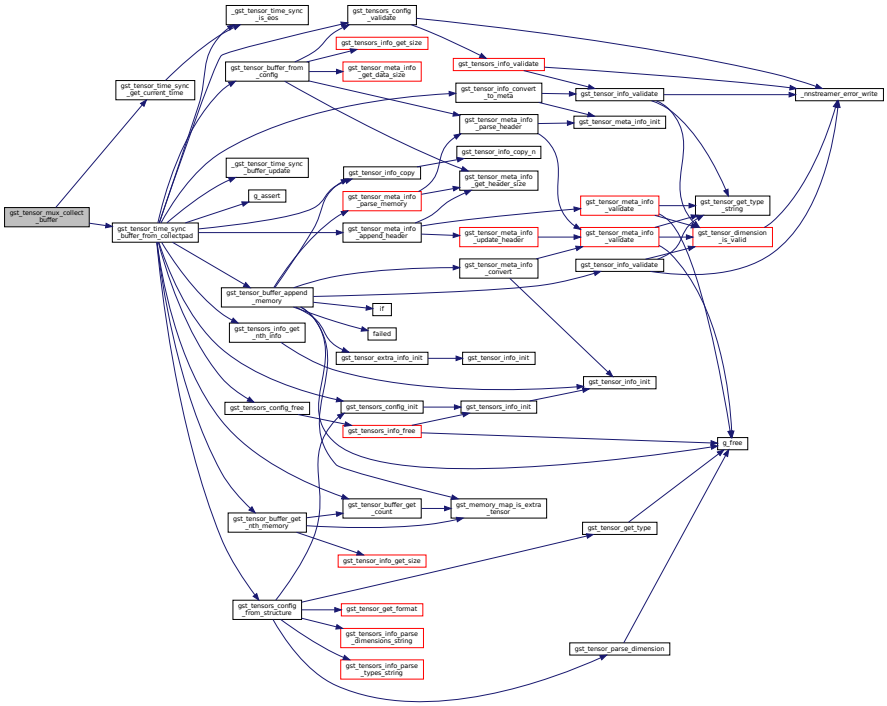
tensor_mux	tensor muxer
tensors_buf	output buffer for srcpad
is_eos	boolean EOS (End of Stream)

Returns

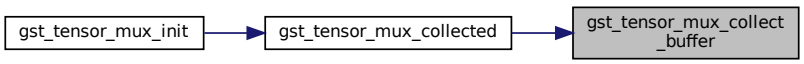
TRUE to push buffer to src pad

Definition at line 368 of file gsttensor_mux.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.44.4.7 gst_tensor_mux_collected()

```
static GstFlowReturn gst_tensor_mux_collected (
            GstCollectPads * pads,
            GstTensorMux * tensor_mux )  [static]
```

Gst Collect Pads Function which is called once collect pads done.

Parameters

pads	GstCollectPads
tensor_mux	Muxer

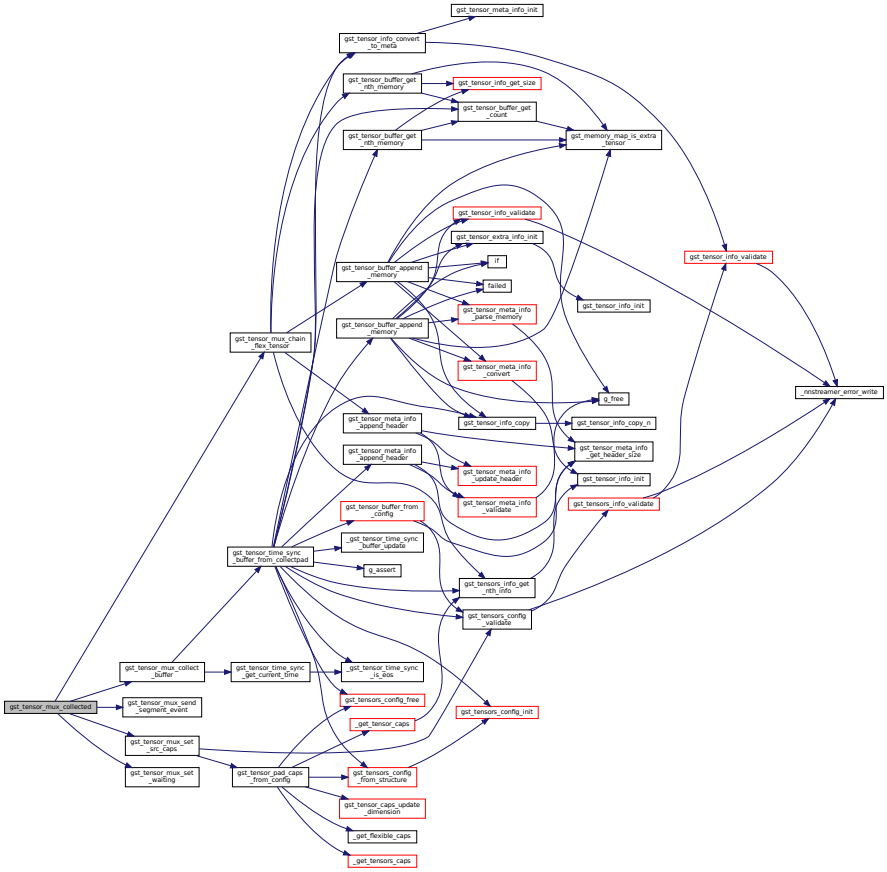
Returns

GstFlowReturn

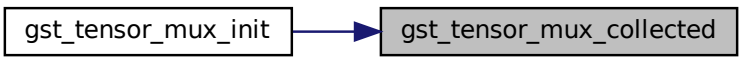
Cannot use gst-pad util to get stream ID (multiple sink pads). Create stream ID using first sink pad.

Definition at line 486 of file gsttensor_mux.c.

Here is the call graph for this function:



Here is the caller graph for this function:



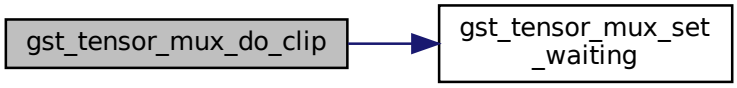
9.44.4.8 gst_tensor_mux_do_clip()

```
static GstFlowReturn gst_tensor_mux_do_clip (
            GstCollectPads * pads,
            GstCollectData * data,
            GstBuffer * buffer,
            GstBuffer ** out,
            GstTensorMux * tensor_mux )  [static]
```

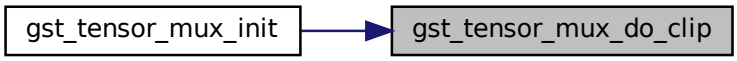
Gst Clip Pads Function which is called right after a buffer is received for each pad.

Definition at line 558 of file gsttensor_mux.c.

Here is the call graph for this function:



Here is the caller graph for this function:



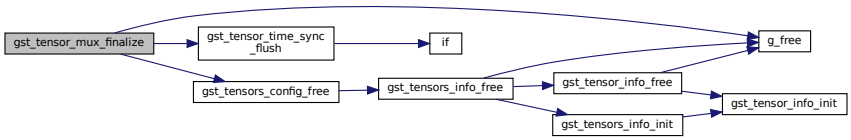
9.44.4.9 gst_tensor_mux_finalize()

```
static void gst_tensor_mux_finalize (
            GObject * object )  [static]
```

finalize vmethod

Definition at line 227 of file gsttensor_mux.c.

Here is the call graph for this function:



Here is the caller graph for this function:

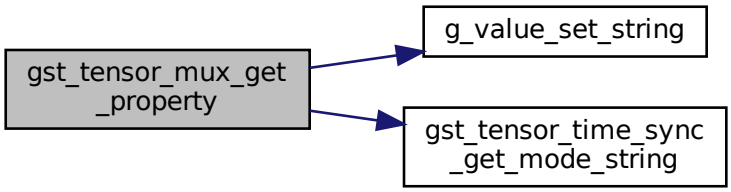
9.44.4.10 gst_tensor_mux_get_property()

```
static void gst_tensor_mux_get_property (
             GObject * object,
             guint prop_id,
             GValue * value,
             GParamSpec * pspec )  [static]
```

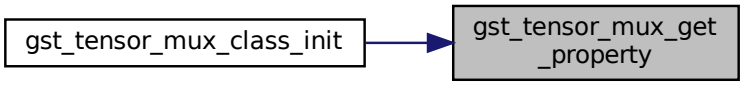
Get property (gst element vmethod)

Definition at line 650 of file gsttensor_mux.c.

Here is the call graph for this function:



Here is the caller graph for this function:



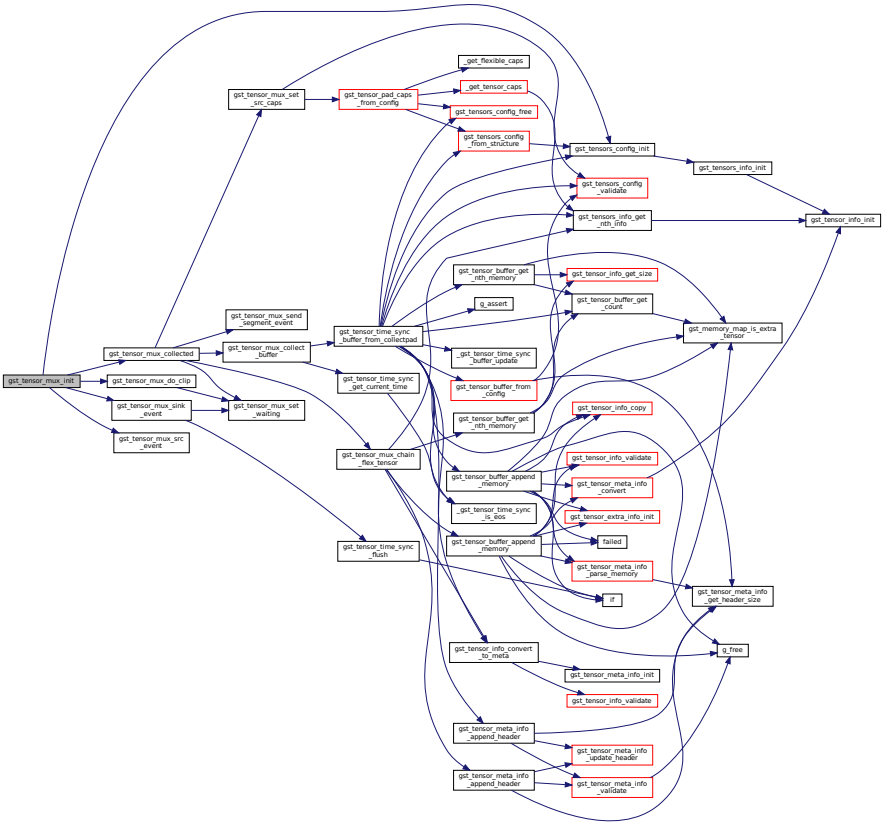
9.44.4.11 gst_tensor_mux_init()

```
static void gst_tensor_mux_init (
            GstTensorMux * tensor_mux )  [static]
```

initialize the new element instantiate pads and add them to element set pad callback functions initialize instance structure

Definition at line 193 of file gsttensor_mux.c.

Here is the call graph for this function:



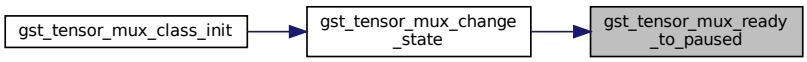
9.44.4.12 gst_tensor_mux_ready_to_paused()

```
static void gst_tensor_mux_ready_to_paused (
            GstTensorMux * tensor_mux )  [static]
```

Ready --> Pasuse State Change.

Definition at line 572 of file gsttensor_mux.c.

Here is the caller graph for this function:



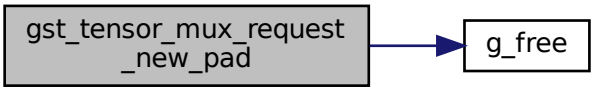
9.44.4.13 gst_tensor_mux_request_new_pad()

```
static GstPad * gst_tensor_mux_request_new_pad (
            GstElement * element,
            GstPadTemplate * templ,
            const gchar * name,
            const GstCaps * caps )  [static]
```

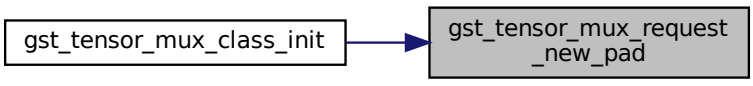
making new request pad (gst element vmethod)

Definition at line 253 of file gsttensor_mux.c.

Here is the call graph for this function:



Here is the caller graph for this function:



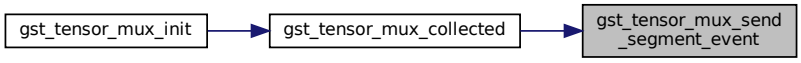
9.44.4.14 gst_tensor_mux_send_segment_event()

```
static void gst_tensor_mux_send_segment_event (
            GstTensorMux * tensor_mux,
            GstClockTime pts,
            GstClockTime dts )  [static]
```

Create a new segment event if necessary.

Definition at line 422 of file gsttensor_mux.c.

Here is the caller graph for this function:



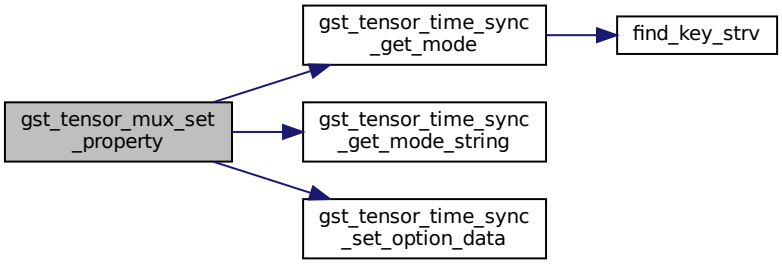
9.44.4.15 gst_tensor_mux_set_property()

```
static void gst_tensor_mux_set_property (
            GObject * object,
            guint prop_id,
            const GValue * value,
            GParamSpec * pspec )  [static]
```

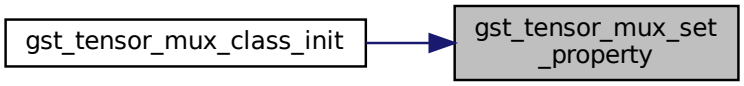
Get property (gst element vmethod)

Definition at line 617 of file gsttensor_mux.c.

Here is the call graph for this function:



Here is the caller graph for this function:



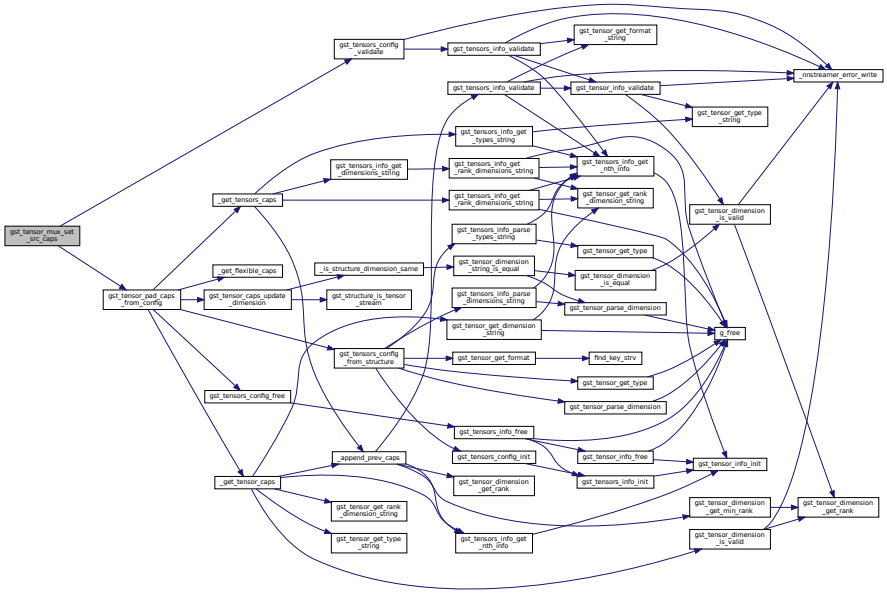
9.44.4.16 gst_tensor_mux_set_src_caps()

```
static gboolean gst_tensor_mux_set_src_caps (
            GstTensorMux * tensor_mux )  [static]
```

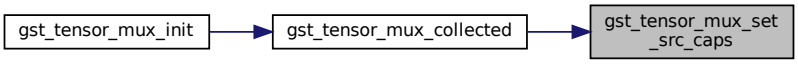
Set src pad caps if src pad is not negotiated.

Definition at line 393 of file gsttensor_mux.c.

Here is the call graph for this function:



Here is the caller graph for this function:



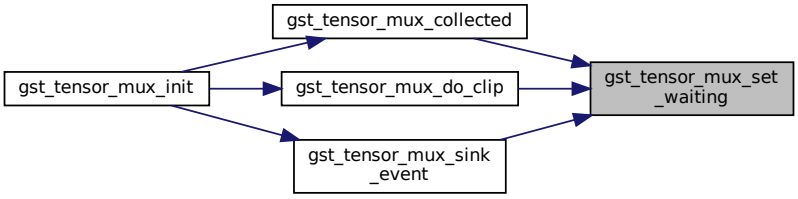
9.44.4.17 gst_tensor_mux_set_waiting()

```
static void gst_tensor_mux_set_waiting (
            GstTensorMux * tensor_mux,
            gboolean waiting )  [static]
```

set pads waiting property

Definition at line 322 of file gsttensor_mux.c.

Here is the caller graph for this function:



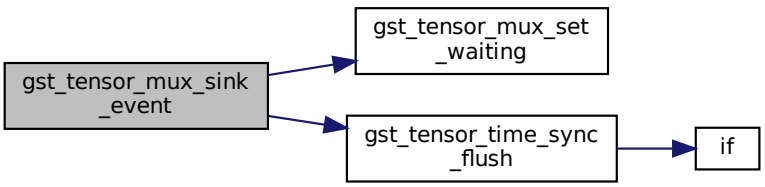
9.44.4.18 gst_tensor_mux_sink_event()

```
static gboolean gst_tensor_mux_sink_event (
            GstCollectPads * pads,
            GstCollectData * data,
            GstEvent * event,
            GstTensorMux * tensor_mux )  [static]
```

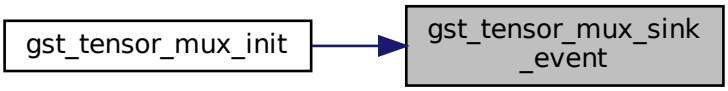
sink event vmethod

Definition at line 339 of file gsttensor_mux.c.

Here is the call graph for this function:



Here is the caller graph for this function:



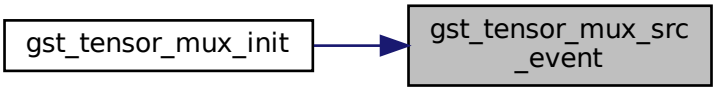
9.44.4.19 gst_tensor_mux_src_event()

```
static gboolean gst_tensor_mux_src_event (
            GstPad * pad,
            GstObject * parent,
            GstEvent * event )  [static]
```

src event vmethod

Definition at line 303 of file gsttensor_mux.c.

Here is the caller graph for this function:



9.44.5 Variable Documentation

9.44.5.1 sink_templ

```
GstStaticPadTemplate sink_templ  [static]
```

Initial value:

```
= GST_STATIC_PAD_TEMPLATE ("sink_%u",
    GST_PAD_SINK,
    GST_PAD_REQUEST,
    )
```

Definition at line 106 of file gsttensor_mux.c.

9.44.5.2 src_templ

```
GstStaticPadTemplate src_templ  [static]
```

Initial value:

```
= GST_STATIC_PAD_TEMPLATE ("src",
    GST_PAD_SRC,
    GST_PAD_ALWAYS,
    )
```

the capabilities of the inputs and outputs. describe the real formats here.

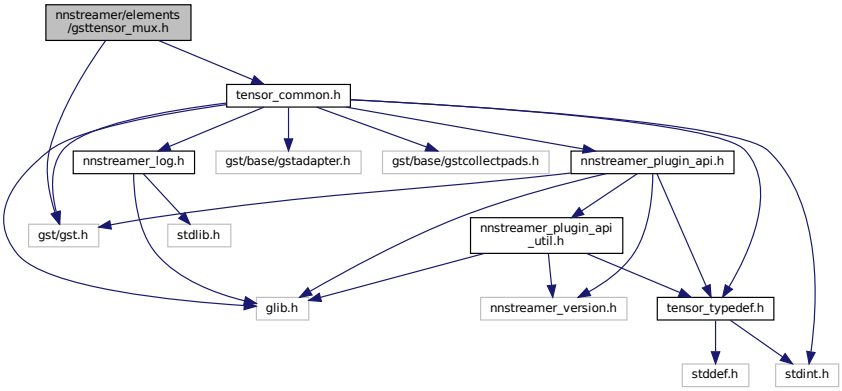
Definition at line 100 of file gsttensor_mux.c.

9.45 nnstreamer/elements/gsttensor_mux.h File Reference

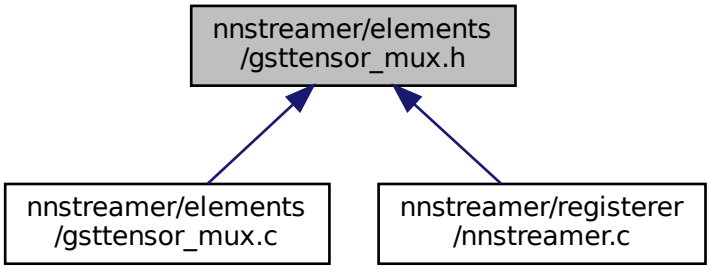
GStreamer plugin to mux tensors (as a filter for other general neural network filters)

```
#include <gst/gst.h>
#include <tensor_common.h>
```

Include dependency graph for gsttensor_mux.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct _GstTensorMux

  Tensor Muxer data structure.
- struct _GstTensorMuxClass

  GstTensroMuxClass inherits GstElementClass.

Macros

- #define GST_TYPE_TENSOR_MUX (gst_tensor_mux_get_type ())
- #define GST_TENSOR_MUX(obj) (G_TYPE_CHECK_INSTANCE_CAST ((obj), GST_TYPE_TENSOR_MUX, GstTensorMux))
- #define GST_TENSOR_MUX_CLASS(klass) (G_TYPE_CHECK_CLASS_CAST ((klass), GST_TYPE_TENSOR_MUX, GstTensorMuxClass))
- #define GST_TENSOR_MUX_GET_CLASS(obj) (G_TYPE_INSTANCE_GET_CLASS ((obj), GST_TYPE_TENSOR_MUX, GstTensorMuxClass))
- #define GST_IS_TENSOR_MUX(obj) (G_TYPE_CHECK_INSTANCE_TYPE((obj),GST_TYPE_TENSOR_MUX))
- #define GST_IS_TENSOR_MUX_CLASS(klass) (G_TYPE_CHECK_CLASS_TYPE((klass),GST_TYPE_TENSOR_MUX))
- #define GST_TENSOR_MUX_CAST(obj) ((GstTensorMux*)(obj))

Typedefs

- typedef struct _GstTensorMux GstTensorMux
- typedef struct _GstTensorMuxClass GstTensorMuxClass

Functions

- GType gst_tensor_mux_get_type (void)

  Get Type function required for gst elements.

9.45.1 Detailed Description

GStreamer plugin to mux tensors (as a filter for other general neural network filters)

GStreamer Copyright (C) 2005 Thomas Vander Stichele thomas@apestaart.org Copyright (C) 2005 Ronald S. Bultje rbultje@ronald.bitfreak.net Copyright (C) 2018 Jijoong Moon jijoong.moon@samsung.com

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Library General Public License as published by the Free Software Foundation; version 2.1 of the License.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Library General Public License for more details.

Date

03 July 2018

See also

https://github.com/nnstreamer/nnstreamer

Author

Jijoong Moon jijoong.moon@samsung.com

Bug No known bugs except for NYI items

9.45.2 Macro Definition Documentation

9.45.2.1 GST_IS_TENSOR_MUX

```
#define GST_IS_TENSOR_MUX(
              obj ) (G_TYPE_CHECK_INSTANCE_TYPE((obj),GST_TYPE_TENSOR_MUX))
```

Definition at line 39 of file gsttensor_mux.h.

9.45.2.2 GST_IS_TENSOR_MUX_CLASS

```
#define GST_IS_TENSOR_MUX_CLASS(
              klass ) (G_TYPE_CHECK_CLASS_TYPE((klass),GST_TYPE_TENSOR_MUX))
```

Definition at line 40 of file gsttensor_mux.h.

9.45.2.3 GST_TENSOR_MUX

```
#define GST_TENSOR_MUX(
              obj ) (G_TYPE_CHECK_INSTANCE_CAST ((obj), GST_TYPE_TENSOR_MUX, GstTensorMux))
```

Definition at line 36 of file gsttensor_mux.h.

9.45.2.4 GST_TENSOR_MUX_CAST

```
#define GST_TENSOR_MUX_CAST(
              obj ) ((GstTensorMux*)(obj))
```

Definition at line 41 of file gsttensor_mux.h.

9.45.2.5 GST_TENSOR_MUX_CLASS

```
#define GST_TENSOR_MUX_CLASS(
              klass ) (G_TYPE_CHECK_CLASS_CAST ((klass), GST_TYPE_TENSOR_MUX, GstTensorMuxClass))
```

Definition at line 37 of file gsttensor_mux.h.

9.45.2.6 GST_TENSOR_MUX_GET_CLASS

```
#define GST_TENSOR_MUX_GET_CLASS(
              obj ) (G_TYPE_INSTANCE_GET_CLASS ((obj), GST_TYPE_TENSOR_MUX, GstTensorMuxClass))
```

Definition at line 38 of file gsttensor_mux.h.

9.45.2.7 GST_TYPE_TENSOR_MUX

```
#define GST_TYPE_TENSOR_MUX (gst_tensor_mux_get_type ())
```

Definition at line 35 of file gsttensor_mux.h.

9.45.3 Typedef Documentation

9.45.3.1 GstTensorMux

```
typedef struct _GstTensorMux GstTensorMux
```

Definition at line 42 of file gsttensor_mux.h.

9.45.3.2 GstTensorMuxClass

```
typedef struct _GstTensorMuxClass GstTensorMuxClass
```

Definition at line 43 of file gsttensor_mux.h.

9.45.4 Function Documentation

9.45.4.1 gst_tensor_mux_get_type()

```
GType gst_tensor_mux_get_type (
             void  )
```

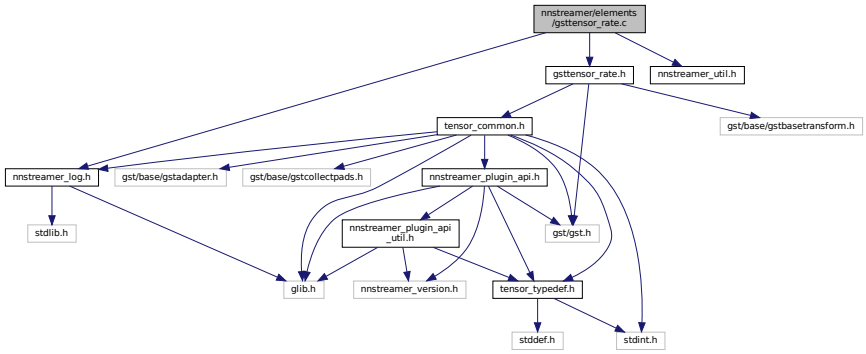
Get Type function required for gst elements.

9.46 nnstreamer/elements/gsttensor_rate.c File Reference

GStreamer plugin to adjust tensor rate.

```
#include <nnstreamer_log.h>
#include <nnstreamer_util.h>
#include "gsttensor_rate.h"
```

Include dependency graph for gsttensor_rate.c:



Macros

- #define DBG (!self->silent)

  Macro for debug mode.
- #define ABSDIFF(a, b) (((a) > (b)) ? (a) - (b) : (b) - (a))
- #define GST_CAT_DEFAULT gst_tensor_rate_debug
- #define CAPS_STRING GST_TENSOR_CAP_DEFAULT "; " GST_TENSORS_CAP_DEFAULT
- #define GST_TENSOR_RATE_SCALED_TIME(self, count)
- #define DEFAULT_SILENT TRUE

  default parameters
- #define DEFAULT_THROTTLE TRUE
- #define gst_tensor_rate_parent_class parent_class
- #define THROTTLE_DELAY_RATIO (0.999)
- #define MAGIC_LIMIT 25

Enumerations

- enum {
  PROP_0, PROP_IN, PROP_OUT, PROP_DUP,
  PROP_DROP, PROP_SILENT, PROP_THROTTLE, PROP_FRAMERATE }

  tensor_rate properties

Functions

- GST_DEBUG_CATEGORY_STATIC (gst_tensor_rate_debug)
- G_DEFINE_TYPE (GstTensorRate, gst_tensor_rate, GST_TYPE_BASE_TRANSFORM)
- static void gst_tensor_rate_set_property (GObject *object, guint prop_id, const GValue *value, GParamSpec *pspec)

  Setter for tensor_rate properties.
- static void gst_tensor_rate_get_property (GObject *object, guint prop_id, GValue *value, GParamSpec *pspec)

  Getter for tensor_rate properties.
- static void gst_tensor_rate_finalize (GObject *object)

  Function to finalize instance. (GST Standard)
- static GstFlowReturn gst_tensor_rate_transform_ip (GstBaseTransform *trans, GstBuffer *buffer)

  in-place transform
- static GstCaps * gst_tensor_rate_transform_caps (GstBaseTransform *trans, GstPadDirection direction, GstCaps *caps, GstCaps *filter)

  configure tensor-srcpad cap from "proposed" cap. (GST Standard)
- static GstCaps * gst_tensor_rate_fixate_caps (GstBaseTransform *trans, GstPadDirection direction, GstCaps *caps, GstCaps *othercaps)

  fixate caps. required vmethod of GstBaseTransform.
- static gboolean gst_tensor_rate_set_caps (GstBaseTransform *trans, GstCaps *in_caps, GstCaps *out_caps)

  set caps. required vmethod of GstBaseTransform.
- static void gst_tensor_rate_swap_prev (GstTensorRate *self, GstBuffer *buffer, gint64 time)

  swap a previous buffer
- static GstFlowReturn gst_tensor_rate_flush_prev (GstTensorRate *self, gboolean duplicate, GstClockTime next_intime)

  flush the oldest buffer
- static void gst_tensor_rate_notify_drop (GstTensorRate *self)

  notify a frame drop event
- static void gst_tensor_rate_notify_duplicate (GstTensorRate *self)

  notify a frame duplicate event
- static gboolean gst_tensor_rate_start (GstBaseTransform *trans)

  Called when the element starts processing. optional vmethod of BaseTransform.
- static gboolean gst_tensor_rate_stop (GstBaseTransform *trans)

  Called when the element stops processing. optional vmethod of BaseTransform.
- static gboolean gst_tensor_rate_sink_event (GstBaseTransform *trans, GstEvent *event)

  Event handler for sink pad of tensor rate.
- static void gst_tensor_rate_install_properties (GObjectClass *object_class)

  Installs all the properties for tensor_rate.
- static void gst_tensor_rate_class_init (GstTensorRateClass *klass)

  initialize the tensor_rate's class (GST Standard)
- static GstFlowReturn gst_tensor_rate_push_buffer (GstTensorRate *self, GstBuffer *outbuf, gboolean duplicate, GstClockTime next_intime)

  push the buffer to src pad
- static void gst_tensor_rate_reset (GstTensorRate *self)

  reset variables of the element (GST Standard)
- static void gst_tensor_rate_init (GstTensorRate *self)

  initialize the new element (GST Standard)
- static void gst_tensor_rate_send_qos_throttle (GstTensorRate *self, GstClockTime timestamp)

  send throttling qos event to upstream elements

Variables

- static GstStaticPadTemplate sink_factory

  The capabilities of the inputs.
- static GstStaticPadTemplate src_factory

  The capabilities of the outputs.
- static GParamSpec * pspec_drop = NULL
- static GParamSpec * pspec_duplicate = NULL

9.46.1 Detailed Description

GStreamer plugin to adjust tensor rate.

GStreamer/NNStreamer Tensor-Rate Copyright (C) 2020 Dongju Chae dongju.chae@samsung.com

Date

24 Sep 2020

See also

https://github.com/nnstreamer/nnstreamer

Author

Dongju Chae dongju.chae@samsung.com

Bug No known bugs except for NYI items

9.46.2 Macro Definition Documentation

9.46.2.1 ABSDIFF

```
#define ABSDIFF(
              a,
              b ) (((a) > (b)) ?  (a) - (b) :  (b) - (a))
```

Definition at line 60 of file gsttensor_rate.c.

9.46.2.2 CAPS_STRING

```
#define CAPS_STRING GST_TENSOR_CAP_DEFAULT "; " GST_TENSORS_CAP_DEFAULT
```

Definition at line 66 of file gsttensor_rate.c.

9.46.2.3 DBG

```
#define DBG (!self->silent)
```

Macro for debug mode.

SECTION:element-tensor_rate

This element controls a frame rate of tensor streams in the pipeline.

Basically, this element takes an incoming stream of tensor frames, and produces an adjusted stream that matches the source pad's framerate. The adjustment is performed by dropping and duplicating tensor frames. By default the element will simply negotiate the same framerate on its source and sink pad.

Also, when 'throttle' property is set, it propagates a specified frame-rate to upstream elements by sending qos events, which prevents unnecessary data from upstream elements.

<refsect2> <title>Example launch line with tensor rate</title> gst-launch-1.0 videotestsrc ! video/x-raw,width=640,height=480,framerate=15/1 ! tensor_converter ! tensor_rate framerate=10/1 throttle=true ! tensor↩_decoder mode=direct_video ! videoconvert ! autovideosink </refsect2>

Definition at line 56 of file gsttensor_rate.c.

9.46.2.4 DEFAULT_SILENT

```
#define DEFAULT_SILENT TRUE
```

default parameters

Definition at line 73 of file gsttensor_rate.c.

9.46.2.5 DEFAULT_THROTTLE

```
#define DEFAULT_THROTTLE TRUE
```

Definition at line 74 of file gsttensor_rate.c.

9.46.2.6 GST_CAT_DEFAULT

```
#define GST_CAT_DEFAULT gst_tensor_rate_debug
```

Definition at line 64 of file gsttensor_rate.c.

9.46.2.7 gst_tensor_rate_parent_class

```
#define gst_tensor_rate_parent_class parent_class
```

Definition at line 110 of file gsttensor_rate.c.

9.46.2.8 GST_TENSOR_RATE_SCALED_TIME

```
#define GST_TENSOR_RATE_SCALED_TIME(
              self,
              count )
```

Value:

```
  gst_util_uint64_scale (count,\
      self->to_rate_denominator * GST_SECOND, self->to_rate_numerator)
```

Definition at line 68 of file gsttensor_rate.c.

9.46.2.9 MAGIC_LIMIT

```
#define MAGIC_LIMIT 25
```

Definition at line 773 of file gsttensor_rate.c.

9.46.2.10 THROTTLE_DELAY_RATIO

```
#define THROTTLE_DELAY_RATIO (0.999)
```

Definition at line 449 of file gsttensor_rate.c.

9.46.3 Enumeration Type Documentation

9.46.3.1 anonymous enum

```
anonymous enum
```

tensor_rate properties

Enumerator

PROP_0	
PROP_IN	
PROP_OUT	
PROP_DUP	
PROP_DROP	
PROP_SILENT	
PROP_THROTTLE	
PROP_FRAMERATE	

Definition at line 79 of file gsttensor_rate.c.

9.46.4 Function Documentation

9.46.4.1 G_DEFINE_TYPE()

```
G_DEFINE_TYPE (
            GstTensorRate ,
            gst_tensor_rate ,
            GST_TYPE_BASE_TRANSFORM  )
```

9.46.4.2 GST_DEBUG_CATEGORY_STATIC()

```
GST_DEBUG_CATEGORY_STATIC (
            gst_tensor_rate_debug  )
```

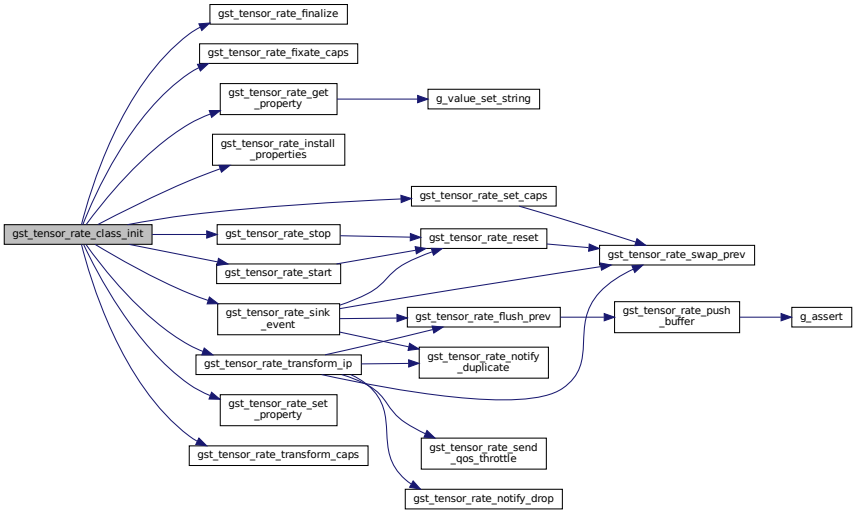
9.46.4.3 gst_tensor_rate_class_init()

```
static void gst_tensor_rate_class_init (
            GstTensorRateClass * klass )  [static]
```

initialize the tensor_rate's class (GST Standard)

Definition at line 148 of file gsttensor_rate.c.

Here is the call graph for this function:



9.46.4.4 gst_tensor_rate_finalize()

```
static void gst_tensor_rate_finalize (
             GObject * object )  [static]
```

Function to finalize instance. (GST Standard)

Definition at line 341 of file gsttensor_rate.c.

Here is the caller graph for this function:

9.46.4.5 gst_tensor_rate_fixate_caps()

```
static GstCaps * gst_tensor_rate_fixate_caps (
            GstBaseTransform * trans,
            GstPadDirection direction,
            GstCaps * caps,
            GstCaps * othercaps )  [static]
```

fixate caps. required vmethod of GstBaseTransform.

Definition at line 680 of file gsttensor_rate.c.

Here is the caller graph for this function:

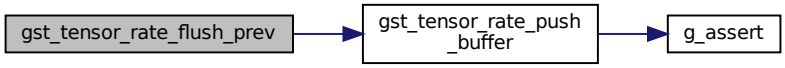
9.46.4.6 gst_tensor_rate_flush_prev()

```
static GstFlowReturn gst_tensor_rate_flush_prev (
            GstTensorRate * self,
            gboolean duplicate,
            GstClockTime next_intime )  [static]
```

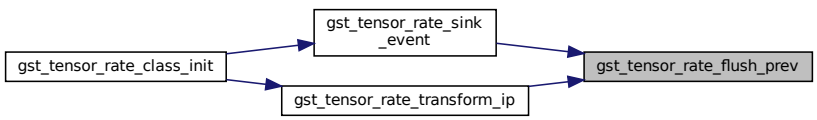
flush the oldest buffer

Definition at line 259 of file gsttensor_rate.c.

Here is the call graph for this function:



Here is the caller graph for this function:



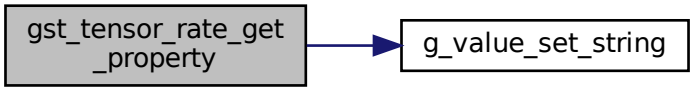
9.46.4.7 gst_tensor_rate_get_property()

```
static void gst_tensor_rate_get_property (
            GObject * object,
            guint prop_id,
            GValue * value,
            GParamSpec * pspec )  [static]
```

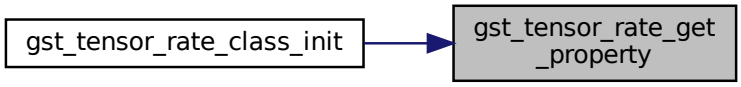
Getter for tensor_rate properties.

Definition at line 407 of file gsttensor_rate.c.

Here is the call graph for this function:



Here is the caller graph for this function:



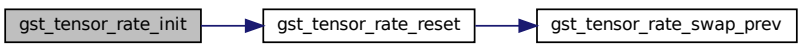
9.46.4.8 gst_tensor_rate_init()

```
static void gst_tensor_rate_init (
            GstTensorRate * self )  [static]
```

initialize the new element (GST Standard)

Definition at line 317 of file gsttensor_rate.c.

Here is the call graph for this function:



9.46.4.9 gst_tensor_rate_install_properties()

```
static void gst_tensor_rate_install_properties (
            GObjectClass * object_class )  [static]
```

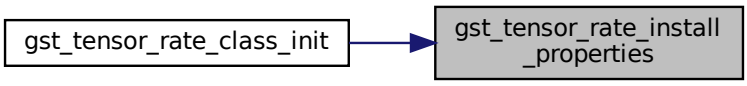
Installs all the properties for tensor_rate.

Parameters

in	*gobject_class*	Glib object class whose properties will be set

Definition at line 953 of file gsttensor_rate.c.

Here is the caller graph for this function:



9.46.4.10 gst_tensor_rate_notify_drop()

```
static void gst_tensor_rate_notify_drop (
            GstTensorRate * self )  [static]
```

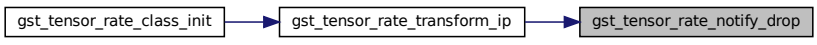
notify a frame drop event

Parameters

in	*self*	"this" pointer

Definition at line 758 of file gsttensor_rate.c.

Here is the caller graph for this function:



9.46.4.11 gst_tensor_rate_notify_duplicate()

```
static void gst_tensor_rate_notify_duplicate (
            GstTensorRate * self )  [static]
```

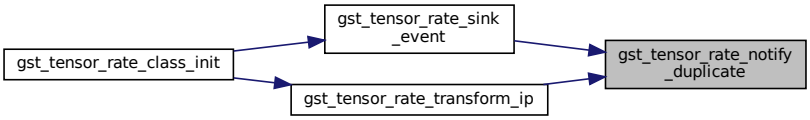
notify a frame duplicate event

Parameters

in	*self*	"this" pointer

Definition at line 768 of file gsttensor_rate.c.

Here is the caller graph for this function:



9.46.4.12 gst_tensor_rate_push_buffer()

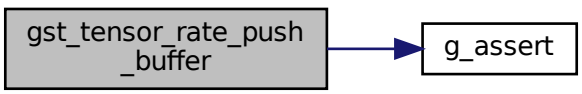
```
static GstFlowReturn gst_tensor_rate_push_buffer (
            GstTensorRate * self,
            GstBuffer * outbuf,
            gboolean duplicate,
            GstClockTime next_intime )  [static]
```

push the buffer to src pad

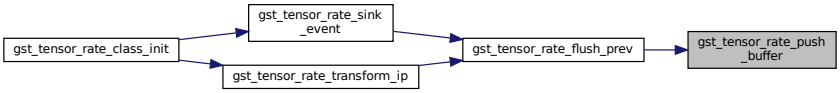
There must always be a valid duration on prevbuf if rate $>$ 0, it is ensured in the transform_ip function

Definition at line 202 of file gsttensor_rate.c.

Here is the call graph for this function:



Here is the caller graph for this function:



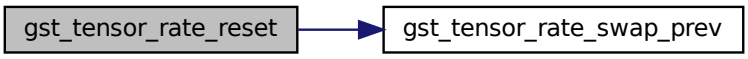
9.46.4.13 gst_tensor_rate_reset()

```
static void gst_tensor_rate_reset (
            GstTensorRate * self )  [static]
```

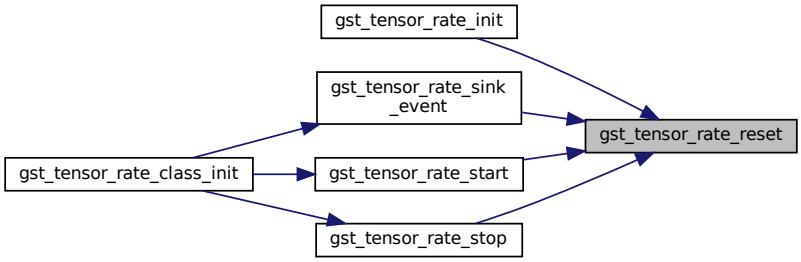
reset variables of the element (GST Standard)

Definition at line 295 of file gsttensor_rate.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.46.4.14 gst_tensor_rate_send_qos_throttle()

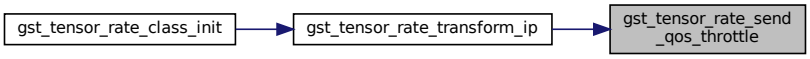
```
static void gst_tensor_rate_send_qos_throttle (
            GstTensorRate * self,
            GstClockTime timestamp )  [static]
```

send throttling qos event to upstream elements

unused

Definition at line 455 of file gsttensor_rate.c.

Here is the caller graph for this function:



9.46.4.15 gst_tensor_rate_set_caps()

```
static gboolean gst_tensor_rate_set_caps (
            GstBaseTransform * trans,
            GstCaps * incaps,
            GstCaps * outcaps )  [static]
```

set caps. required vmethod of GstBaseTransform.

After a setcaps, our caps may have changed. In that case, we can't use the old buffer, if there was one (it might have different dimensions)

Definition at line 705 of file gsttensor_rate.c.

Here is the call graph for this function:

Here is the caller graph for this function:

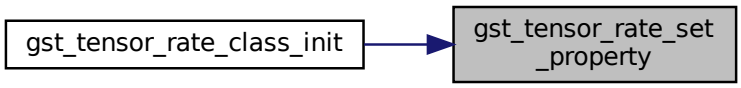
9.46.4.16 gst_tensor_rate_set_property()

```
static void gst_tensor_rate_set_property (
            GObject * object,
            guint prop_id,
            const GValue * value,
            GParamSpec * pspec )  [static]
```

Setter for tensor_rate properties.

Definition at line 350 of file gsttensor_rate.c.

Here is the caller graph for this function:



9.46.4.17 gst_tensor_rate_sink_event()

```
static gboolean gst_tensor_rate_sink_event (
            GstBaseTransform * trans,
            GstEvent * event )  [static]
```

Event handler for sink pad of tensor rate.

Parameters

in	*trans*	"this" pointer
in	*event*	a passed event object

Returns

TRUE if there is no error.

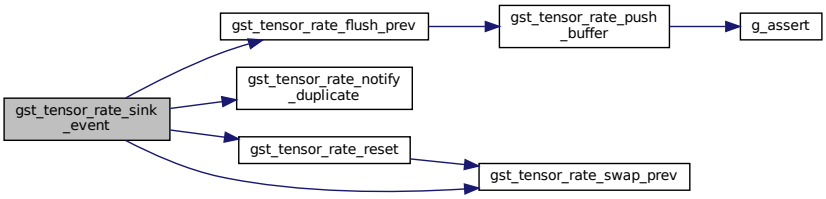
fill up to the end of current segment, or only send out the stored buffer if there is no specific stop. regardless, prevent going loopy in strange cases

fill up to the end of current segment, or only send out the stored buffer if there is no specific stop. regardless, prevent going loopy in strange cases

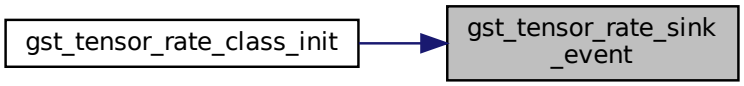
Output at least one frame but if the buffer duration is valid, output enough frames to use the complete buffer duration

Definition at line 781 of file gsttensor_rate.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.46.4.18 gst_tensor_rate_start()

```
static gboolean gst_tensor_rate_start (
            GstBaseTransform * trans )  [static]
```

Called when the element starts processing. optional vmethod of BaseTransform.

Parameters

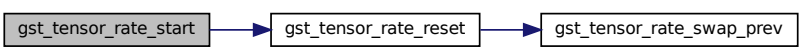
in	*trans*	"this" pointer

Returns

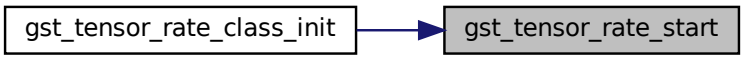
TRUE if there is no error.

Definition at line 928 of file gsttensor_rate.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.46.4.19 gst_tensor_rate_stop()

```
static gboolean gst_tensor_rate_stop (
            GstBaseTransform * trans )  [static]
```

Called when the element stops processing. optional vmethod of BaseTransform.

Parameters

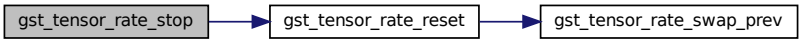
in	*trans*	"this" pointer

Returns

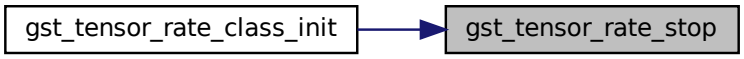
TRUE if there is no error.

Definition at line 941 of file gsttensor_rate.c.

Here is the call graph for this function:



Here is the caller graph for this function:



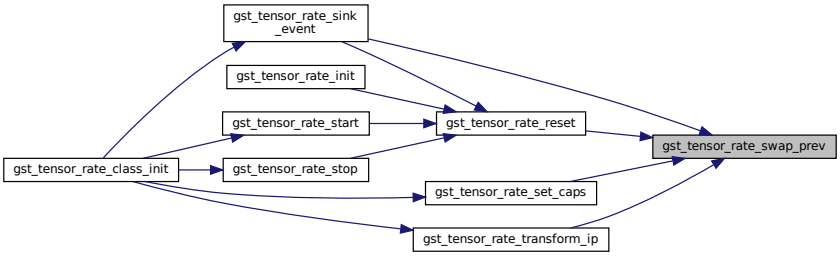
9.46.4.20 gst_tensor_rate_swap_prev()

```
static void gst_tensor_rate_swap_prev (
            GstTensorRate * self,
            GstBuffer * buffer,
            gint64 time )  [static]
```

swap a previous buffer

Definition at line 280 of file gsttensor_rate.c.

Here is the caller graph for this function:



9.46.4.21 gst_tensor_rate_transform_caps()

```
static GstCaps * gst_tensor_rate_transform_caps (
            GstBaseTransform * trans,
            GstPadDirection direction,
            GstCaps * caps,
            GstCaps * filter )  [static]
```

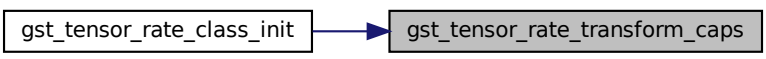
configure tensor-srcpad cap from "proposed" cap. (GST Standard)

@trans ("this" pointer) @direction (why do we need this?) @caps sinkpad cap (if direction GST_PAD_SINK) @filter this element's cap (don't know specifically.)

Be careful not to fix/set caps at this stage. Negotiation not completed yet.

Definition at line 634 of file gsttensor_rate.c.

Here is the caller graph for this function:



9.46.4.22 gst_tensor_rate_transform_ip()

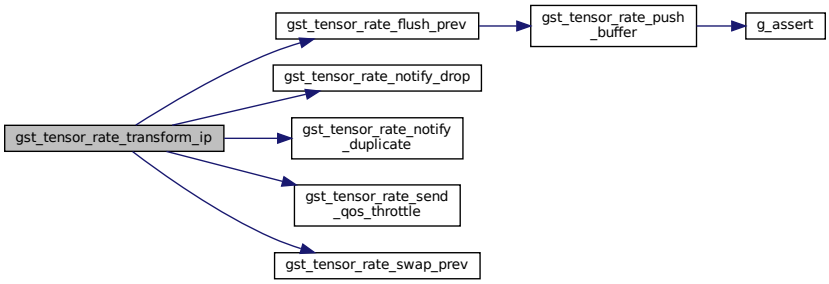
```
static GstFlowReturn gst_tensor_rate_transform_ip (
            GstBaseTransform * trans,
            GstBuffer * buffer )  [static]
```

in-place transform

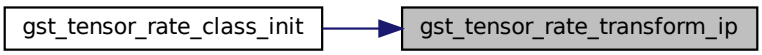
continue while the first one was the best, if they were equal avoid going into an infinite loop

Definition at line 477 of file gsttensor_rate.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.46.5 Variable Documentation

9.46.5.1 pspec_drop

```
GParamSpec* pspec_drop = NULL  [static]
```

Definition at line 107 of file gsttensor_rate.c.

9.46.5.2 pspec_duplicate

```
GParamSpec* pspec_duplicate = NULL  [static]
```

Definition at line 108 of file gsttensor_rate.c.

9.46.5.3 sink_factory

```
GstStaticPadTemplate sink_factory  [static]
```

Initial value:

```
= GST_STATIC_PAD_TEMPLATE ("sink",
    GST_PAD_SINK,
    GST_PAD_ALWAYS,
    GST_STATIC_CAPS (CAPS_STRING))
```

The capabilities of the inputs.

Definition at line 94 of file gsttensor_rate.c.

9.46.5.4 src_factory

```
GstStaticPadTemplate src_factory  [static]
```

Initial value:

```
= GST_STATIC_PAD_TEMPLATE ("src",
    GST_PAD_SRC,
    GST_PAD_ALWAYS,
    GST_STATIC_CAPS (CAPS_STRING))
```

The capabilities of the outputs.

Definition at line 102 of file gsttensor_rate.c.

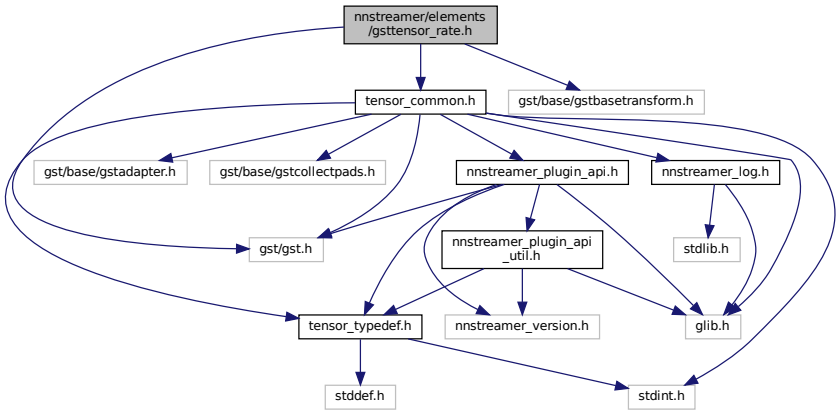
9.47 nnstreamer/elements/gsttensor_rate.h File Reference

GStreamer plugin to adjust tensor rate.

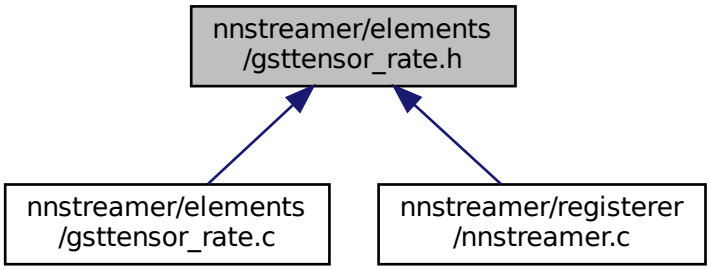
```
#include <gst/gst.h>
#include <gst/base/gstbasetransform.h>
```

```
#include <tensor_common.h>
```

Include dependency graph for gsttensor_rate.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct _GstTensorRate

  Tensor Rate data structure.
- struct _GstTensorRateClass

  GstTensorRateClass inherits GstElementClass.

Macros

- #define GST_TYPE_TENSOR_RATE (gst_tensor_rate_get_type ())
- #define GST_TENSOR_RATE(obj) (G_TYPE_CHECK_INSTANCE_CAST ((obj), GST_TYPE_TENSOR_RATE, GstTensorRate))
- #define GST_TENSOR_RATE_CLASS(klass) (G_TYPE_CHECK_CLASS_CAST ((klass), GST_TYPE_TENSOR_RATE, GstTensorRateClass))

- #define GST_TENSOR_RATE_GET_CLASS(obj) (G_TYPE_INSTANCE_GET_CLASS ((obj), GST_TYPE_TENSOR_RATE, GstTensorRateClass))
- #define GST_IS_TENSOR_RATE(obj) (G_TYPE_CHECK_INSTANCE_TYPE((obj),GST_TYPE_TENSOR_RATE))
- #define GST_IS_TENSOR_RATE_CLASS(klass) (G_TYPE_CHECK_CLASS_TYPE((klass),GST_TYPE_TENSOR_RATE))
- #define GST_TENSOR_RATE_CAST(obj) ((GstTensorRate*)(obj))

Typedefs

- typedef struct _GstTensorRate GstTensorRate
- typedef struct _GstTensorRateClass GstTensorRateClass

Functions

- GType gst_tensor_rate_get_type (void)

  Get Type function required for gst elements.

9.47.1 Detailed Description

GStreamer plugin to adjust tensor rate.

GStreamer/NNStreamer Tensor-Rate Copyright (C) 2020 Dongju Chae dongju.chae@samsung.com

Date

24 Sep 2020

See also

https://github.com/nnstreamer/nnstreamer

Author

Dongju Chae dongju.chae@samsung.com

Bug No known bugs except for NYI items

9.47.2 Macro Definition Documentation

9.47.2.1 GST_IS_TENSOR_RATE

```
#define GST_IS_TENSOR_RATE(
              obj ) (G_TYPE_CHECK_INSTANCE_TYPE((obj),GST_TYPE_TENSOR_RATE))
```

Definition at line 28 of file gsttensor_rate.h.

9.47.2.2 GST_IS_TENSOR_RATE_CLASS

```
#define GST_IS_TENSOR_RATE_CLASS(
              klass ) (G_TYPE_CHECK_CLASS_TYPE((klass),GST_TYPE_TENSOR_RATE))
```

Definition at line 29 of file gsttensor_rate.h.

9.47.2.3 GST_TENSOR_RATE

```
#define GST_TENSOR_RATE(
              obj ) (G_TYPE_CHECK_INSTANCE_CAST ((obj), GST_TYPE_TENSOR_RATE, GstTensorRate))
```

Definition at line 25 of file gsttensor_rate.h.

9.47.2.4 GST_TENSOR_RATE_CAST

```
#define GST_TENSOR_RATE_CAST(
              obj ) ((GstTensorRate*)(obj))
```

Definition at line 30 of file gsttensor_rate.h.

9.47.2.5 GST_TENSOR_RATE_CLASS

```
#define GST_TENSOR_RATE_CLASS(
              klass ) (G_TYPE_CHECK_CLASS_CAST ((klass), GST_TYPE_TENSOR_RATE, GstTensorRateClass))
```

Definition at line 26 of file gsttensor_rate.h.

9.47.2.6 GST_TENSOR_RATE_GET_CLASS

```
#define GST_TENSOR_RATE_GET_CLASS(
              obj ) (G_TYPE_INSTANCE_GET_CLASS ((obj), GST_TYPE_TENSOR_RATE, GstTensorRateClass))
```

Definition at line 27 of file gsttensor_rate.h.

9.47.2.7 GST_TYPE_TENSOR_RATE

```
#define GST_TYPE_TENSOR_RATE (gst_tensor_rate_get_type ())
```

Definition at line 24 of file gsttensor_rate.h.

9.47.3 Typedef Documentation

9.47.3.1 GstTensorRate

```
typedef struct _GstTensorRate GstTensorRate
```

Definition at line 31 of file gsttensor_rate.h.

9.47.3.2 GstTensorRateClass

```
typedef struct _GstTensorRateClass GstTensorRateClass
```

Definition at line 32 of file gsttensor_rate.h.

9.47.4 Function Documentation

9.47.4.1 gst_tensor_rate_get_type()

```
GType gst_tensor_rate_get_type (
            void  )
```

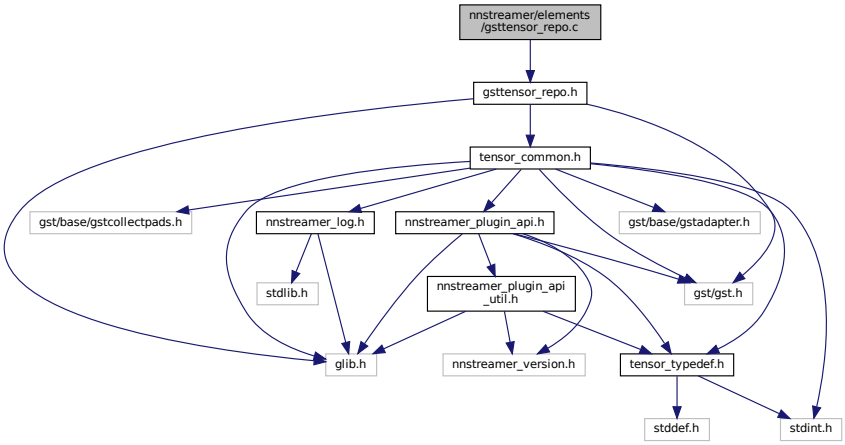
Get Type function required for gst elements.

9.48 nnstreamer/elements/gsttensor_repo.c File Reference

tensor repo file for NNStreamer, the GStreamer plugin for neural networks

```
#include "gsttensor_repo.h"
```

Include dependency graph for gsttensor_repo.c:



Macros

- #define DBG FALSE
- #define GST_REPO_LOCK() (g_mutex_lock(&_repo.repo_lock))

  Macro for Lock & Cond.
- #define GST_REPO_UNLOCK() (g_mutex_unlock(&_repo.repo_lock))
- #define GST_REPO_WAIT() (g_cond_wait(&_repo.repo_cond, &_repo.repo_lock))
- #define GST_REPO_BROADCAST() (g_cond_broadcast (&_repo.repo_cond))

Functions

- static void gst_tensor_repo_release_repodata (gpointer data)

  Internal function to release repo data.
- GstTensorRepoData * gst_tensor_repo_get_repodata (guint nth)

  Getter to get nth GstTensorRepoData.
- gboolean gst_tensor_repo_set_changed (guint o_nth, guint nth, gboolean is_sink)

  Set the changing status of repo.
- gboolean gst_tensor_repo_add_repodata (guint nth, gboolean is_sink)

  Add GstTensorRepoData into repo.
- gboolean gst_tensor_repo_set_buffer (guint nth, GstBuffer *buffer, GstCaps *caps)

  Push GstBuffer into repo.
- gboolean gst_tensor_repo_check_eos (guint nth)

  Check EOS (End-of-Stream) of slot.
- gboolean gst_tensor_repo_check_changed (guint nth, guint *newid, gboolean is_sink)

  Check repo data is changed.
- gboolean gst_tensor_repo_set_eos (guint nth)

  Set EOS (End-of-Stream) of slot.
- GstBuffer * gst_tensor_repo_get_buffer (guint nth, gboolean *eos, guint *newid, GstCaps **caps)

  Get GstTensorRepoData from repo.
- gboolean gst_tensor_repo_remove_repodata (guint nth)

  Remove nth GstTensorRepoData from GstTensorRepo.
- void gst_tensor_repo_init (void)

  GstTensorRepo initialization.
- gboolean gst_tensor_repo_wait (void)

  Wait for finish of initialization.

Variables

- static GstTensorRepo _repo = {.num_data = 0,.initialized = FALSE }

  tensor repo global variable with init.

9.48.1 Detailed Description

tensor repo file for NNStreamer, the GStreamer plugin for neural networks

Date

17 Nov 2018

See also

https://github.com/nnstreamer/nnstreamer

Author

Jijoong Moon jijoong.moon@samsung.com

Bug No known bugs except for NYI items

9.48.2 Macro Definition Documentation

9.48.2.1 DBG

```
#define DBG FALSE
```

Definition at line 29 of file gsttensor_repo.c.

9.48.2.2 GST_REPO_BROADCAST

```
#define GST_REPO_BROADCAST( ) (g_cond_broadcast (&_repo.repo_cond))
```

Definition at line 43 of file gsttensor_repo.c.

9.48.2.3 GST_REPO_LOCK

```
#define GST_REPO_LOCK( ) (g_mutex_lock(&_repo.repo_lock))
```

Macro for Lock & Cond.

Definition at line 40 of file gsttensor_repo.c.

9.48.2.4 GST_REPO_UNLOCK

```
#define GST_REPO_UNLOCK( ) (g_mutex_unlock(&_repo.repo_lock))
```

Definition at line 41 of file gsttensor_repo.c.

9.48.2.5 GST_REPO_WAIT

```
#define GST_REPO_WAIT( ) (g_cond_wait(&_repo.repo_cond, &_repo.repo_lock))
```

Definition at line 42 of file gsttensor_repo.c.

9.48.3 Function Documentation

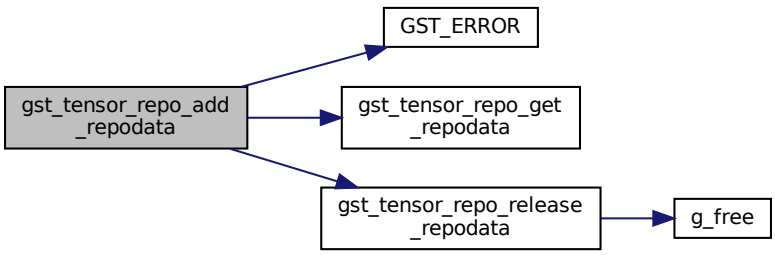
9.48.3.1 gst_tensor_repo_add_repodata()

```
gboolean gst_tensor_repo_add_repodata (
            guint nth,
            gboolean is_sink )
```

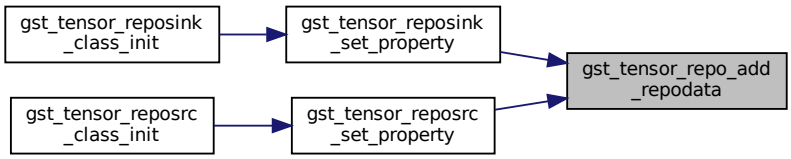
Add GstTensorRepoData into repo.

Definition at line 129 of file gsttensor_repo.c.

Here is the call graph for this function:



Here is the caller graph for this function:



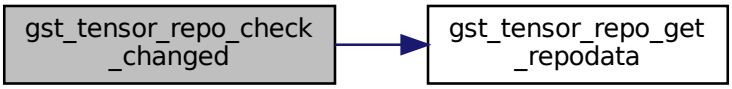
9.48.3.2 gst_tensor_repo_check_changed()

```
gboolean gst_tensor_repo_check_changed (
            guint nth,
            guint * newid,
            gboolean is_sink )
```

Check repo data is changed.

Definition at line 255 of file gsttensor_repo.c.

Here is the call graph for this function:



Here is the caller graph for this function:

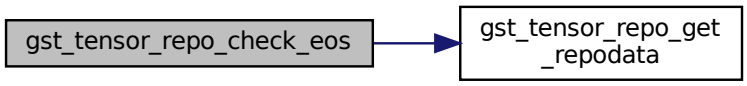
9.48.3.3 gst_tensor_repo_check_eos()

```
gboolean gst_tensor_repo_check_eos (
            guint nth )
```

Check EOS (End-of-Stream) of slot.

Definition at line 236 of file gsttensor_repo.c.

Here is the call graph for this function:



Here is the caller graph for this function:

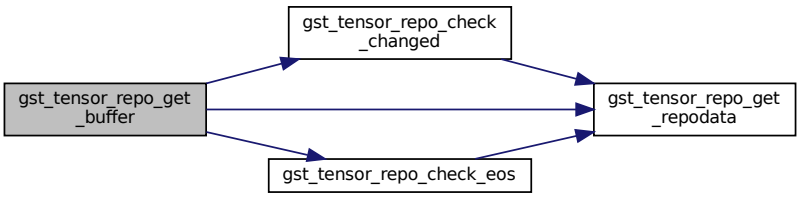
9.48.3.4 gst_tensor_repo_get_buffer()

```
GstBuffer* gst_tensor_repo_get_buffer (
            guint nth,
            gboolean * eos,
            guint * newid,
            GstCaps ** caps )
```

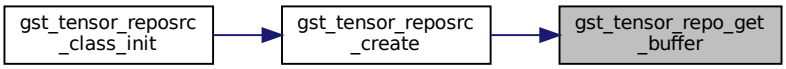
Get GstTensorRepoData from repo.

Definition at line 309 of file gsttensor_repo.c.

Here is the call graph for this function:



Here is the caller graph for this function:



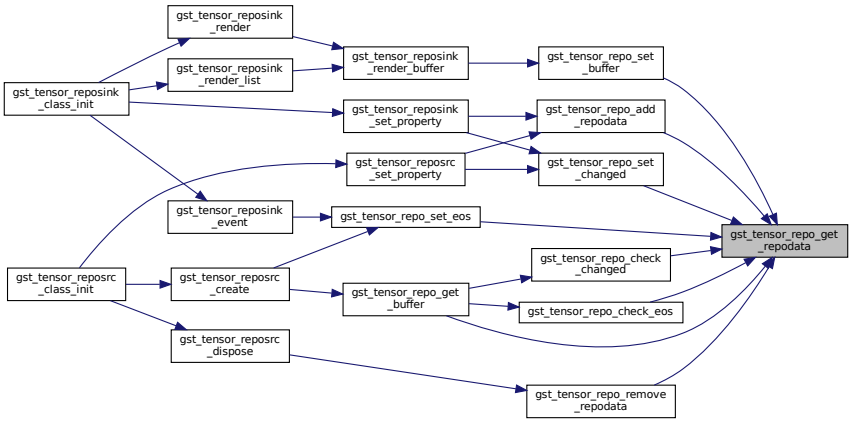
9.48.3.5 gst_tensor_repo_get_repodata()

```
GstTensorRepoData* gst_tensor_repo_get_repodata (
            guint nth )
```

Getter to get nth GstTensorRepoData.

Definition at line 74 of file gsttensor_repo.c.

Here is the caller graph for this function:



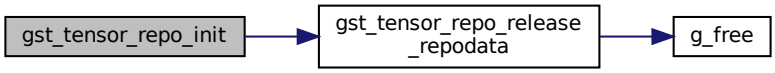
9.48.3.6 gst_tensor_repo_init()

```
void gst_tensor_repo_init (
            void  )
```

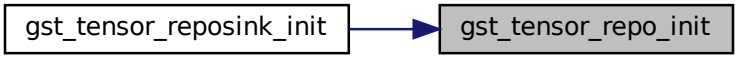
GstTensorRepo initialization.

Definition at line 386 of file gsttensor_repo.c.

Here is the call graph for this function:



Here is the caller graph for this function:



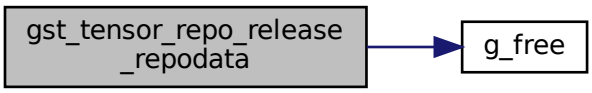
9.48.3.7 gst_tensor_repo_release_repodata()

```
static void gst_tensor_repo_release_repodata (
            gpointer data )  [static]
```

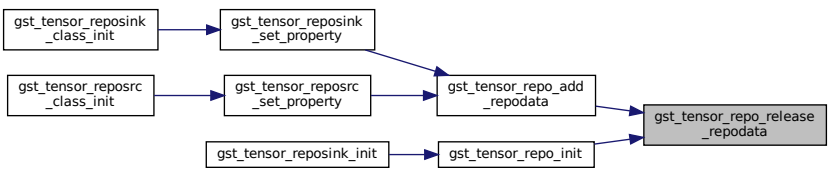
Internal function to release repo data.

Definition at line 49 of file gsttensor_repo.c.

Here is the call graph for this function:



Here is the caller graph for this function:



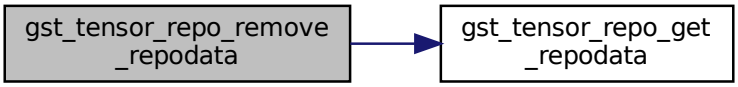
9.48.3.8 gst_tensor_repo_remove_repodata()

```
gboolean gst_tensor_repo_remove_repodata (
            guint nth )
```

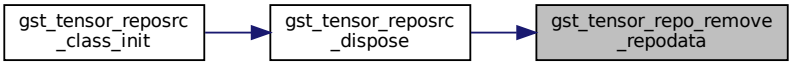
Remove nth GstTensorRepoData from GstTensorRepo.

Definition at line 357 of file gsttensor_repo.c.

Here is the call graph for this function:



Here is the caller graph for this function:



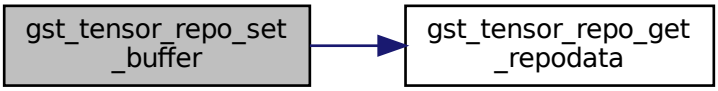
9.48.3.9 gst_tensor_repo_set_buffer()

```
gboolean gst_tensor_repo_set_buffer (
            guint nth,
            GstBuffer * buffer,
            GstCaps * caps )
```

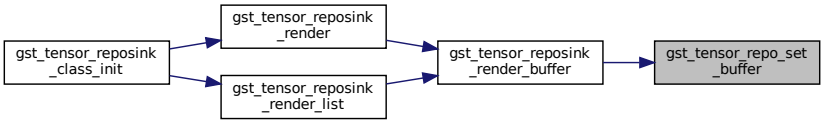
Push GstBuffer into repo.

Definition at line 193 of file gsttensor_repo.c.

Here is the call graph for this function:



Here is the caller graph for this function:



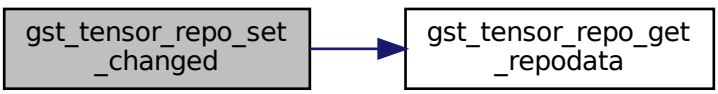
9.48.3.10 gst_tensor_repo_set_changed()

```
gboolean gst_tensor_repo_set_changed (
            guint o_nth,
            guint nth,
            gboolean is_sink )
```

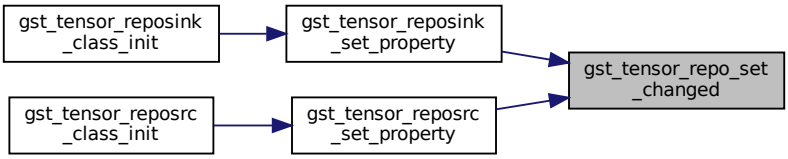
Set the changing status of repo.

Definition at line 91 of file gsttensor_repo.c.

Here is the call graph for this function:



Here is the caller graph for this function:



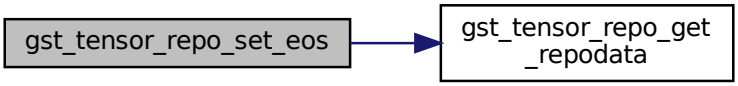
9.48.3.11 gst_tensor_repo_set_eos()

```
gboolean gst_tensor_repo_set_eos (
            guint nth )
```

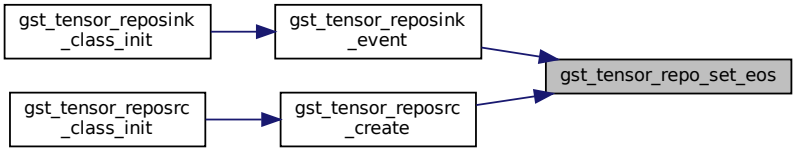
Set EOS (End-of-Stream) of slot.

Definition at line 287 of file gsttensor_repo.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.48.3.12 gst_tensor_repo_wait()

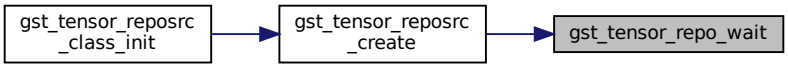
```
gboolean gst_tensor_repo_wait (
            void  )
```

Wait for finish of initialization.

Wait for the repo initialization.

Definition at line 406 of file gsttensor_repo.c.

Here is the caller graph for this function:



9.48.4 Variable Documentation

9.48.4.1 _repo

```
GstTensorRepo _repo = {.num_data = 0,.initialized = FALSE }  [static]
```

tensor repo global variable with init.

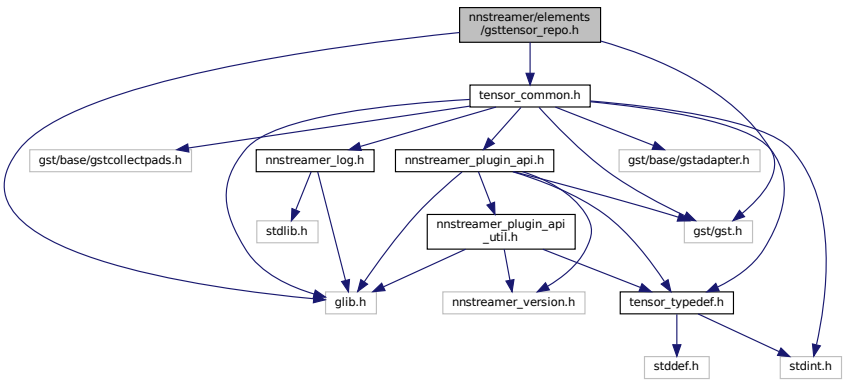
Definition at line 35 of file gsttensor_repo.c.

9.49 nnstreamer/elements/gsttensor_repo.h File Reference

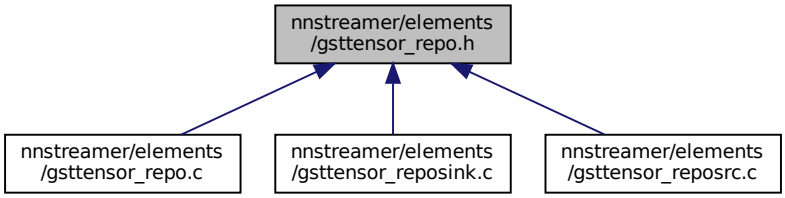
tensor repo header file for NNStreamer, the GStreamer plugin for neural networks

```
#include <glib.h>
#include <gst/gst.h>
#include "tensor_common.h"
```

Include dependency graph for gsttensor_repo.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct GstTensorRepoData

  GstTensorRepo internal data structure.
- struct GstTensorRepo

  GstTensorRepo data structure.

Functions

- GstTensorRepoData ∗ gst_tensor_repo_get_repodata (guint nth)

  Getter to get nth GstTensorRepoData.
- gboolean gst_tensor_repo_add_repodata (guint myid, gboolean is_sink)

  Add GstTensorRepoData into repo.
- gboolean gst_tensor_repo_set_buffer (guint nth, GstBuffer ∗buffer, GstCaps ∗caps)

  Push GstBuffer into repo.
- gboolean gst_tensor_repo_check_eos (guint nth)

  Check EOS (End-of-Stream) of slot.
- gboolean gst_tensor_repo_set_eos (guint nth)

  Set EOS (End-of-Stream) of slot.
- gboolean gst_tensor_repo_set_changed (guint o_nth, guint nth, gboolean is_sink)

  Set the changing status of repo.
- GstBuffer ∗ gst_tensor_repo_get_buffer (guint nth, gboolean ∗eos, guint ∗newid, GstCaps ∗∗caps)

  Get GstTensorRepoData from repo.
- gboolean gst_tensor_repo_check_changed (guint nth, guint ∗newid, gboolean is_sink)

  Check repo data is changed.
- gboolean gst_tensor_repo_remove_repodata (guint nth)

  Remove nth GstTensorRepoData from GstTensorRepo.
- void gst_tensor_repo_init (void)

  GstTensorRepo initialization.
- gboolean gst_tensor_repo_wait (void)

  Wait for the repo initialization.

9.49.1 Detailed Description

tensor repo header file for NNStreamer, the GStreamer plugin for neural networks

NNStreamer Tensor Repo Header Copyright (C) 2018 Jijoong Moon jijoong.moon@samsung.com

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Library General Public License as published by the Free Software Foundation; version 2.1 of the License.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Library General Public License for more details.

Date

17 Nov 2018

See also

https://github.com/nnstreamer/nnstreamer

Author

Jijoong Moon jijoong.moon@samsung.com

Bug No known bugs except for NYI items

9.49.2 Function Documentation

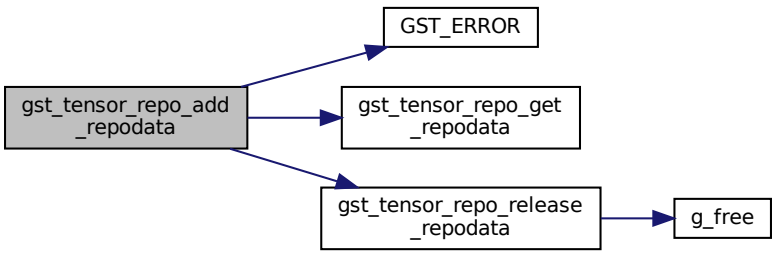
9.49.2.1 gst_tensor_repo_add_repodata()

```
gboolean gst_tensor_repo_add_repodata (
            guint myid,
            gboolean is_sink )
```

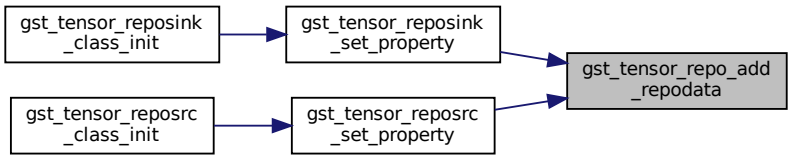
Add GstTensorRepoData into repo.

Definition at line 129 of file gsttensor_repo.c.

Here is the call graph for this function:



Here is the caller graph for this function:



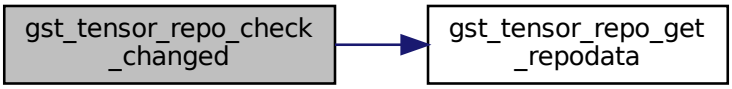
9.49.2.2 gst_tensor_repo_check_changed()

```
gboolean gst_tensor_repo_check_changed (
            guint nth,
            guint * newid,
            gboolean is_sink )
```

Check repo data is changed.

Definition at line 255 of file gsttensor_repo.c.

Here is the call graph for this function:



Here is the caller graph for this function:

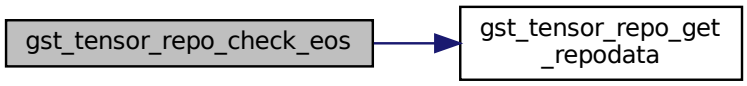
9.49.2.3 gst_tensor_repo_check_eos()

```
gboolean gst_tensor_repo_check_eos (
            guint nth )
```

Check EOS (End-of-Stream) of slot.

Definition at line 236 of file gsttensor_repo.c.

Here is the call graph for this function:



Here is the caller graph for this function:

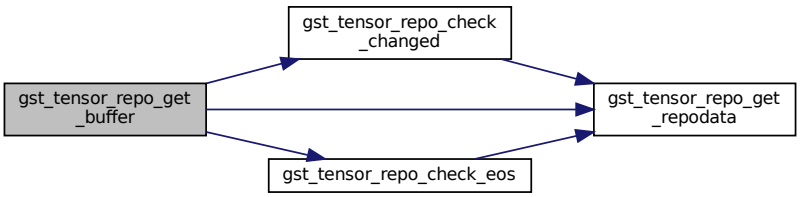
9.49.2.4 gst_tensor_repo_get_buffer()

```
GstBuffer* gst_tensor_repo_get_buffer (
            guint nth,
            gboolean * eos,
            guint * newid,
            GstCaps ** caps )
```

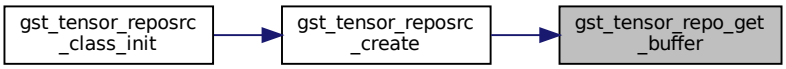
Get GstTensorRepoData from repo.

Definition at line 309 of file gsttensor_repo.c.

Here is the call graph for this function:



Here is the caller graph for this function:



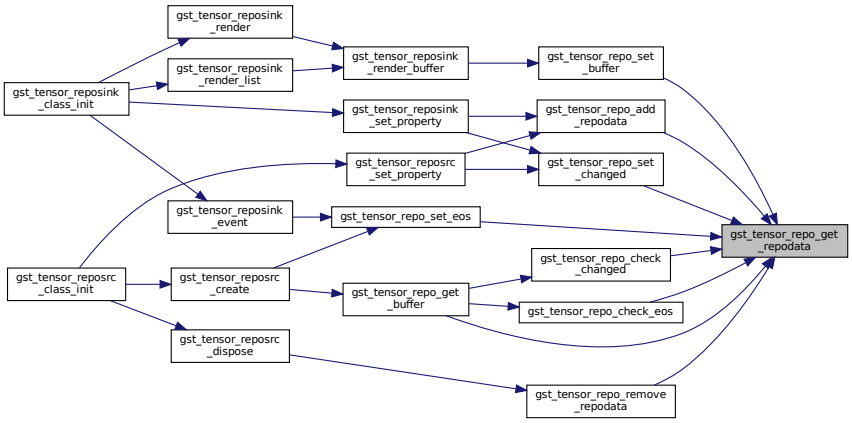
9.49.2.5 gst_tensor_repo_get_repodata()

```
GstTensorRepoData* gst_tensor_repo_get_repodata (
            guint nth )
```

Getter to get nth GstTensorRepoData.

Definition at line 74 of file gsttensor_repo.c.

Here is the caller graph for this function:



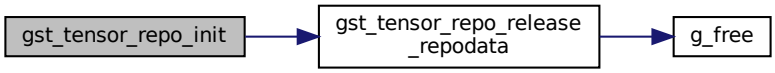
9.49.2.6 gst_tensor_repo_init()

```
void gst_tensor_repo_init (
            void  )
```

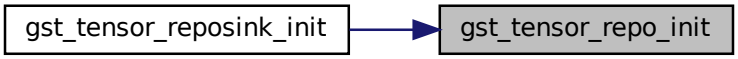
GstTensorRepo initialization.

Definition at line 386 of file gsttensor_repo.c.

Here is the call graph for this function:



Here is the caller graph for this function:



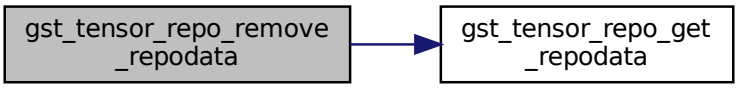
9.49.2.7 gst_tensor_repo_remove_repodata()

```
gboolean gst_tensor_repo_remove_repodata (
            guint nth )
```

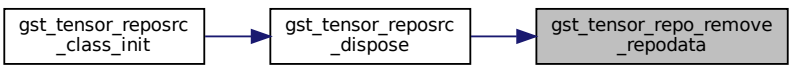
Remove nth GstTensorRepoData from GstTensorRepo.

Definition at line 357 of file gsttensor_repo.c.

Here is the call graph for this function:



Here is the caller graph for this function:



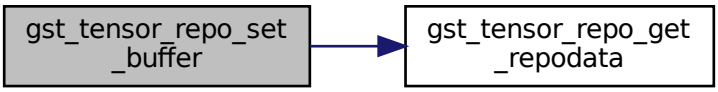
9.49.2.8 gst_tensor_repo_set_buffer()

```
gboolean gst_tensor_repo_set_buffer (
            guint nth,
            GstBuffer * buffer,
            GstCaps * caps )
```

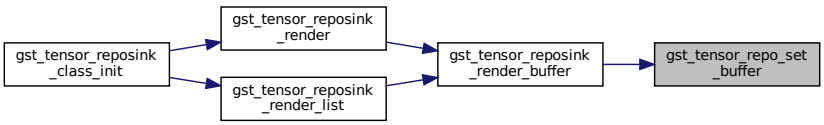
Push GstBuffer into repo.

Definition at line 193 of file gsttensor_repo.c.

Here is the call graph for this function:



Here is the caller graph for this function:



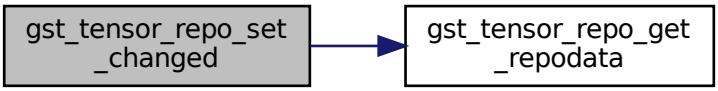
9.49.2.9 gst_tensor_repo_set_changed()

```
gboolean gst_tensor_repo_set_changed (
            guint o_nth,
            guint nth,
            gboolean is_sink )
```

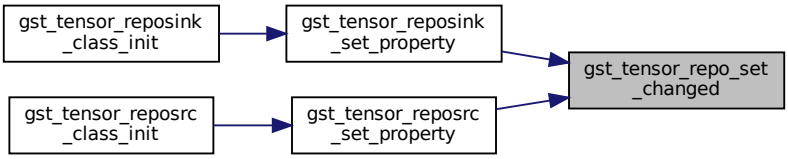
Set the changing status of repo.

Definition at line 91 of file gsttensor_repo.c.

Here is the call graph for this function:



Here is the caller graph for this function:



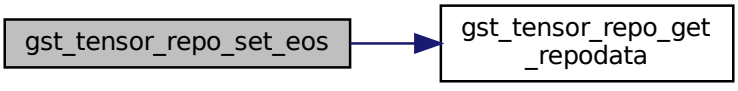
9.49.2.10 gst_tensor_repo_set_eos()

```
gboolean gst_tensor_repo_set_eos (
             guint nth )
```

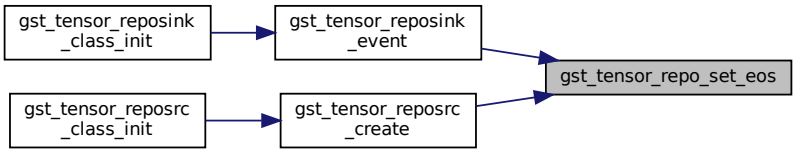
Set EOS (End-of-Stream) of slot.

Definition at line 287 of file gsttensor_repo.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.49.2.11 gst_tensor_repo_wait()

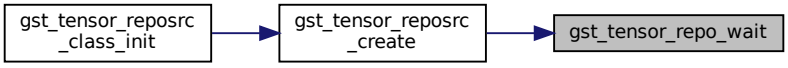
```
gboolean gst_tensor_repo_wait (
            void  )
```

Wait for the repo initialization.

Wait for the repo initialization.

Definition at line 406 of file gsttensor_repo.c.

Here is the caller graph for this function:

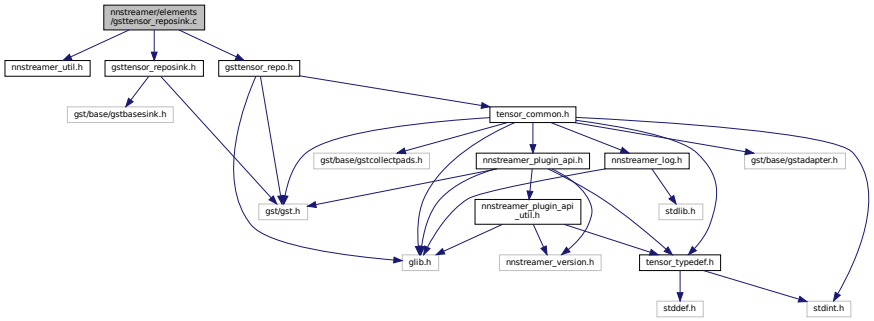


9.50 nnstreamer/elements/gsttensor_reposink.c File Reference

GStreamer plugin to handle tensor repository.

```
#include <nnstreamer_util.h>
#include "gsttensor_repo.h"
#include "gsttensor_reposink.h"
```

Include dependency graph for gsttensor_reposink.c:



Macros

- #define GST_CAT_DEFAULT gst_tensor_reposink_debug
- #define DEFAULT_SIGNAL_RATE 0
- #define DEFAULT_SILENT TRUE
- #define DEFAULT_QOS TRUE
- #define DEFAULT_INDEX 0
- #define gst_tensor_reposink_parent_class parent_class

Enumerations

- enum { PROP_0, PROP_SIGNAL_RATE, PROP_SLOT, PROP_SILENT }

  tensor_reposink properties

Functions

- GST_DEBUG_CATEGORY_STATIC (gst_tensor_reposink_debug)
- static void gst_tensor_reposink_set_property (GObject *object, guint prop_id, const GValue *value, GParamSpec *pspec)

  set property vmethod
- static void gst_tensor_reposink_get_property (GObject *object, guint prop_id, GValue *value, GParamSpec *pspec)

  get property vmethod
- static void gst_tensor_reposink_dispose (GObject *object)

  dispose vmethod implementation
- static gboolean gst_tensor_reposink_start (GstBaseSink *sink)

  start vmethod implementation
- static gboolean gst_tensor_reposink_stop (GstBaseSink *sink)

  stop vmethod implementation
- static gboolean gst_tensor_reposink_event (GstBaseSink *sink, GstEvent *event)

  Handle events.
- static gboolean gst_tensor_reposink_query (GstBaseSink *sink, GstQuery *query)

  query vmethod implementation
- static GstFlowReturn gst_tensor_reposink_render (GstBaseSink *sink, GstBuffer *buffer)

  render vmethod implementation
- static GstFlowReturn gst_tensor_reposink_render_list (GstBaseSink *sink, GstBufferList *buffer_list)

  render list vmethod implementation
- static gboolean gst_tensor_reposink_set_caps (GstBaseSink *sink, GstCaps *caps)

  set_caps vmethod implementation
- static GstCaps * gst_tensor_reposink_get_caps (GstBaseSink *sink, GstCaps *filter)

  get_caps vmethod implementation
- G_DEFINE_TYPE (GstTensorRepoSink, gst_tensor_reposink, GST_TYPE_BASE_SINK)
- static void gst_tensor_reposink_class_init (GstTensorRepoSinkClass *klass)

  class initialization of tensor_reposink
- static void gst_tensor_reposink_init (GstTensorRepoSink *self)

  initialization of tensor_reposink
- static gboolean gst_tensor_reposink_render_buffer (GstTensorRepoSink *self, GstBuffer *buffer)

  Push GstBuffer.

9.50.1 Detailed Description

GStreamer plugin to handle tensor repository.

GStreamer Copyright (C) 2005 Thomas Vander Stichele thomas@apestaart.org Copyright (C) 2005 Ronald S. Bultje rbultje@ronald.bitfreak.net Copyright (C) 2018 Samsung Electronics Co., Ltd.

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Library General Public License as published by the Free Software Foundation; version 2.1 of the License.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Library General Public License for more details. SECTION: element-tensor_reposink

Set element to handle tensor repo

Date

19 Nov 2018

See also

https://github.com/nnstreamer/nnstreamer

Author

Jijoong Moon jijoong.moon@samsung.com

Bug No known bugs except for NYI items

9.50.2 Macro Definition Documentation

9.50.2.1 DEFAULT_INDEX

```
#define DEFAULT_INDEX 0
```

Definition at line 56 of file gsttensor_reposink.c.

9.50.2.2 DEFAULT_QOS

```
#define DEFAULT_QOS TRUE
```

Definition at line 55 of file gsttensor_reposink.c.

9.50.2.3 DEFAULT_SIGNAL_RATE

```
#define DEFAULT_SIGNAL_RATE 0
```

Definition at line 53 of file gsttensor_reposink.c.

9.50.2.4 DEFAULT_SILENT

```
#define DEFAULT_SILENT TRUE
```

Definition at line 54 of file gsttensor_reposink.c.

9.50.2.5 GST_CAT_DEFAULT

```
#define GST_CAT_DEFAULT gst_tensor_reposink_debug
```

Definition at line 40 of file gsttensor_reposink.c.

9.50.2.6 gst_tensor_reposink_parent_class

```
#define gst_tensor_reposink_parent_class parent_class
```

Definition at line 79 of file gsttensor_reposink.c.

9.50.3 Enumeration Type Documentation

9.50.3.1 anonymous enum

```
anonymous enum
```

tensor_reposink properties

Enumerator

PROP_0	
PROP_SIGNAL_RATE	
PROP_SLOT	
PROP_SILENT	

Definition at line 45 of file gsttensor_reposink.c.

9.50.4 Function Documentation

9.50.4.1 G_DEFINE_TYPE()

```
G_DEFINE_TYPE (
            GstTensorRepoSink ,
            gst_tensor_reposink ,
            GST_TYPE_BASE_SINK  )
```

9.50.4.2 GST_DEBUG_CATEGORY_STATIC()

```
GST_DEBUG_CATEGORY_STATIC (
            gst_tensor_reposink_debug  )
```

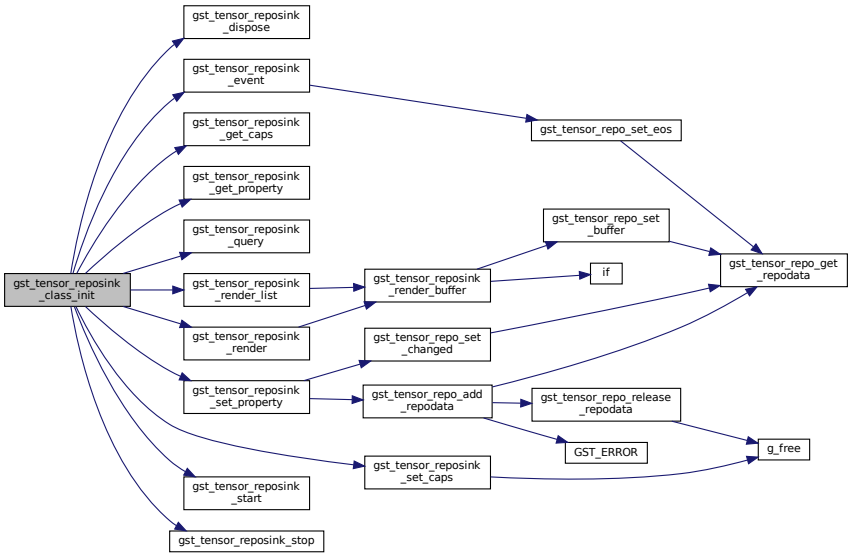
9.50.4.3 gst_tensor_reposink_class_init()

```
static void gst_tensor_reposink_class_init (
            GstTensorRepoSinkClass * klass )  [static]
```

class initialization of tensor_reposink

Definition at line 86 of file gsttensor_reposink.c.

Here is the call graph for this function:



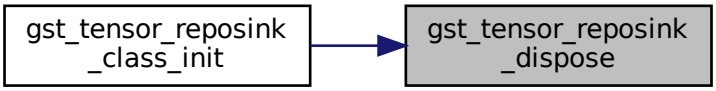
9.50.4.4 gst_tensor_reposink_dispose()

```
static void gst_tensor_reposink_dispose (
            GObject * object )  [static]
```

dispose vmethod implementation

Definition at line 240 of file gsttensor_reposink.c.

Here is the caller graph for this function:



9.50.4.5 gst_tensor_reposink_event()

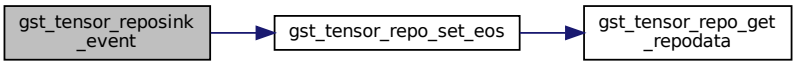
```
static gboolean gst_tensor_reposink_event (
            GstBaseSink * sink,
            GstEvent * event )  [static]
```

Handle events.

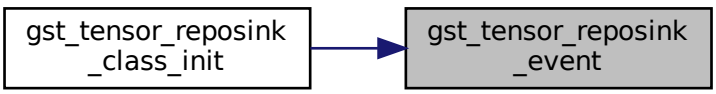
GstBaseSink method implementation.

Definition at line 278 of file gsttensor_reposink.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.50.4.6 gst_tensor_reposink_get_caps()

```
static GstCaps * gst_tensor_reposink_get_caps (
            GstBaseSink * sink,
            GstCaps * filter )  [static]
```

get_caps vmethod implementation

Definition at line 448 of file gsttensor_reposink.c.

Here is the caller graph for this function:

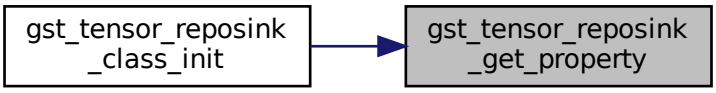
9.50.4.7 gst_tensor_reposink_get_property()

```
static void gst_tensor_reposink_get_property (
            GObject * object,
            guint prop_id,
            GValue * value,
            GParamSpec * pspec )  [static]
```

get property vmethod

Definition at line 213 of file gsttensor_reposink.c.

Here is the caller graph for this function:



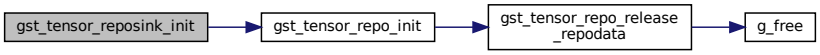
9.50.4.8 gst_tensor_reposink_init()

```
static void gst_tensor_reposink_init (
            GstTensorRepoSink * self )  [static]
```

initialization of tensor_reposink

Definition at line 148 of file gsttensor_reposink.c.

Here is the call graph for this function:



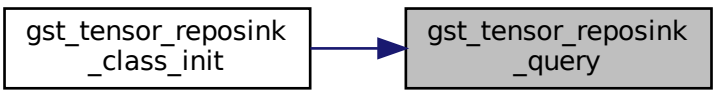
9.50.4.9 gst_tensor_reposink_query()

```
static gboolean gst_tensor_reposink_query (
            GstBaseSink * sink,
            GstQuery * query )  [static]
```

query vmethod implementation

Definition at line 303 of file gsttensor_reposink.c.

Here is the caller graph for this function:



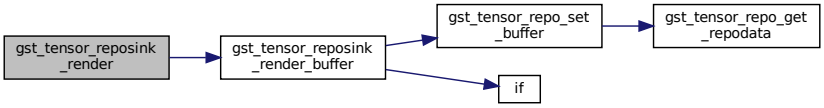
9.50.4.10 gst_tensor_reposink_render()

```
static GstFlowReturn gst_tensor_reposink_render (
            GstBaseSink * sink,
            GstBuffer * buffer )  [static]
```

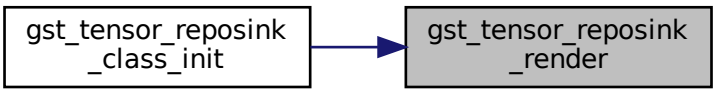
render vmethod implementation

Definition at line 377 of file gsttensor_reposink.c.

Here is the call graph for this function:



Here is the caller graph for this function:



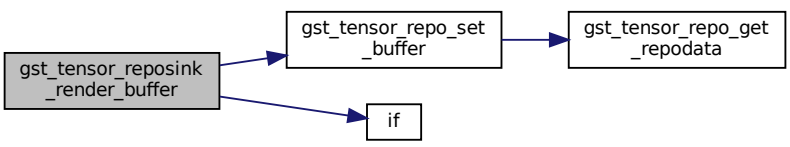
9.50.4.11 gst_tensor_reposink_render_buffer()

```
static gboolean gst_tensor_reposink_render_buffer (
            GstTensorRepoSink * self,
            GstBuffer * buffer )  [static]
```

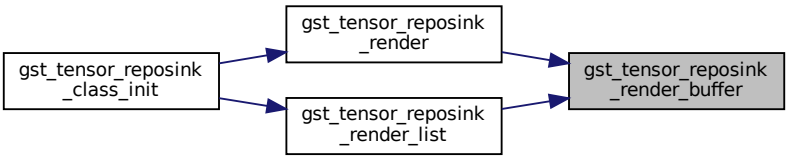
Push GstBuffer.

Definition at line 330 of file gsttensor_reposink.c.

Here is the call graph for this function:



Here is the caller graph for this function:



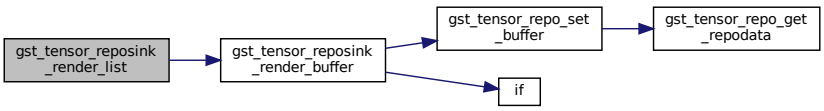
9.50.4.12 gst_tensor_reposink_render_list()

```
static GstFlowReturn gst_tensor_reposink_render_list (
            GstBaseSink * sink,
            GstBufferList * buffer_list )  [static]
```

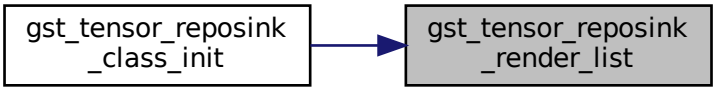
render list vmethod implementation

Definition at line 392 of file gsttensor_reposink.c.

Here is the call graph for this function:



Here is the caller graph for this function:



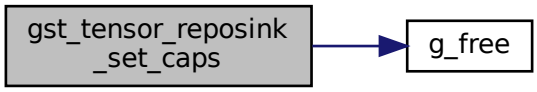
9.50.4.13 gst_tensor_reposink_set_caps()

```
static gboolean gst_tensor_reposink_set_caps (
            GstBaseSink * sink,
            GstCaps * caps )  [static]
```

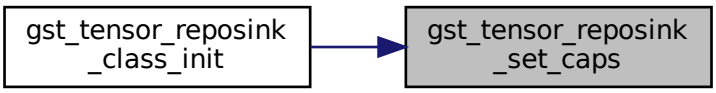
set_caps vmethod implementation

Definition at line 416 of file gsttensor_reposink.c.

Here is the call graph for this function:



Here is the caller graph for this function:



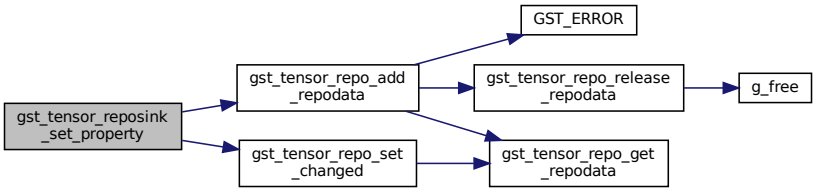
9.50.4.14 gst_tensor_reposink_set_property()

```
static void gst_tensor_reposink_set_property (
            GObject * object,
            guint prop_id,
            const GValue * value,
            GParamSpec * pspec )  [static]
```

set property vmethod

Definition at line 175 of file gsttensor_reposink.c.

Here is the call graph for this function:



Here is the caller graph for this function:

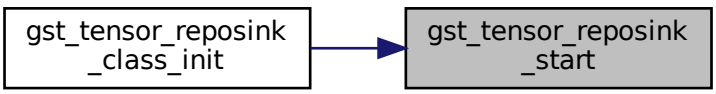
9.50.4.15 gst_tensor_reposink_start()

```
static gboolean gst_tensor_reposink_start (
            GstBaseSink * sink )  [static]
```

start vmethod implementation

Definition at line 256 of file gsttensor_reposink.c.

Here is the caller graph for this function:



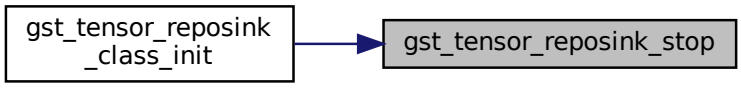
9.50.4.16 gst_tensor_reposink_stop()

```
static gboolean gst_tensor_reposink_stop (
            GstBaseSink * sink )  [static]
```

stop vmethod implementation

Definition at line 266 of file gsttensor_reposink.c.

Here is the caller graph for this function:

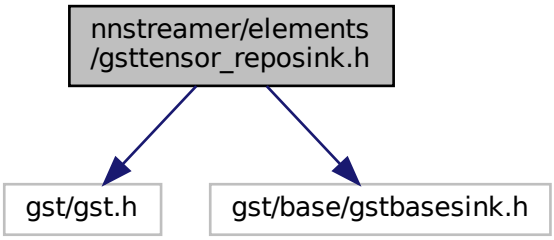


9.51 nnstreamer/elements/gsttensor_reposink.h File Reference

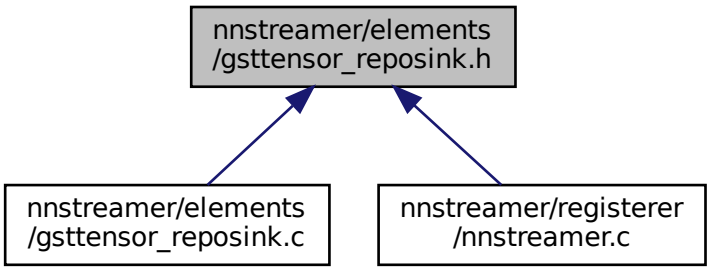
GStreamer plugin to handle tensor repository.

```
#include <gst/gst.h>
#include <gst/base/gstbasesink.h>
```

Include dependency graph for gsttensor_reposink.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct _GstTensorRepoSink

  GstTensorRepoSink data structure.
- struct _GstTensorRepoSinkClass

  GstTensorRepoSinkClass data structure.

Macros

- #define GST_TYPE_TENSOR_REPOSINK (gst_tensor_reposink_get_type())
- #define GST_TENSOR_REPOSINK(obj) (G_TYPE_CHECK_INSTANCE_CAST((obj),GST_TYPE_TENSOR_REPOSINK,GstT
- #define GST_TENSOR_REPOSINK_CLASS(klass) (G_TYPE_CHECK_CLASS_CAST((klass),GST_TYPE_TENSOR_REPOS
- #define GST_IS_TENSOR_REPOSINK(obj) (G_TYPE_CHECK_INSTANCE_TYPE((obj),GST_TYPE_TENSOR_REPOSINK))
- #define GST_IS_TENSOR_REPOSINK_CLASS(klass) (G_TYPE_CHECK_CLASS_TYPE((klass),GST_TYPE_TENSOR_REP

Typedefs

- typedef struct _GstTensorRepoSink GstTensorRepoSink
- typedef struct _GstTensorRepoSinkClass GstTensorRepoSinkClass

Functions

- GType gst_tensor_reposink_get_type (void)

  Function to get type of tensor_reposink.

9.51.1 Detailed Description

GStreamer plugin to handle tensor repository.

GStreamer Copyright (C) 2005 Thomas Vander Stichele thomas@apestaart.org Copyright (C) 2005 Ronald S. Bultje rbultje@ronald.bitfreak.net Copyright (C) 2018 Samsung Electronics Co., Ltd.

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Library General Public License as published by the Free Software Foundation; version 2.1 of the License.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Library General Public License for more details.

Date

19 Nov 2018

See also

https://github.com/nnstreamer/nnstreamer

Author

Jijoong Moon jijoong.moon@samsung.com

Bug No known bugs except for NYI items

9.51.2 Macro Definition Documentation

9.51.2.1 GST_IS_TENSOR_REPOSINK

```
#define GST_IS_TENSOR_REPOSINK(
              obj ) (G_TYPE_CHECK_INSTANCE_TYPE((obj),GST_TYPE_TENSOR_REPOSINK))
```

Definition at line 41 of file gsttensor_reposink.h.

9.51.2.2 GST_IS_TENSOR_REPOSINK_CLASS

```
#define GST_IS_TENSOR_REPOSINK_CLASS(
              klass ) (G_TYPE_CHECK_CLASS_TYPE((klass),GST_TYPE_TENSOR_REPOSINK))
```

Definition at line 43 of file gsttensor_reposink.h.

9.51.2.3 GST_TENSOR_REPOSINK

```
#define GST_TENSOR_REPOSINK(
              obj ) (G_TYPE_CHECK_INSTANCE_CAST((obj),GST_TYPE_TENSOR_REPOSINK,GstTensorRepoSink))
```

Definition at line 37 of file gsttensor_reposink.h.

9.51.2.4 GST_TENSOR_REPOSINK_CLASS

```
#define GST_TENSOR_REPOSINK_CLASS(
              klass ) (G_TYPE_CHECK_CLASS_CAST((klass),GST_TYPE_TENSOR_REPOSINK,GstTensorRepoSinkClass))
```

Definition at line 39 of file gsttensor_reposink.h.

9.51.2.5 GST_TYPE_TENSOR_REPOSINK

```
#define GST_TYPE_TENSOR_REPOSINK (gst_tensor_reposink_get_type())
```

Definition at line 35 of file gsttensor_reposink.h.

9.51.3 Typedef Documentation

9.51.3.1 GstTensorRepoSink

```
typedef struct _GstTensorRepoSink GstTensorRepoSink
```

Definition at line 46 of file gsttensor_reposink.h.

9.51.3.2 GstTensorRepoSinkClass

```
typedef struct _GstTensorRepoSinkClass GstTensorRepoSinkClass
```

Definition at line 47 of file gsttensor_reposink.h.

9.51.4 Function Documentation

9.51.4.1 gst_tensor_reposink_get_type()

```
GType gst_tensor_reposink_get_type (
            void  )
```

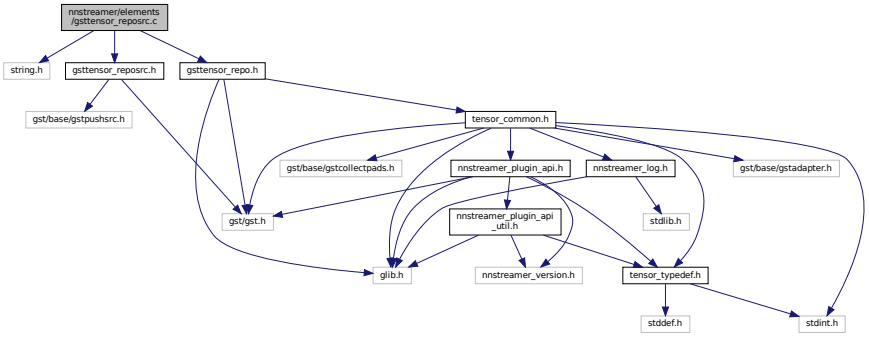
Function to get type of tensor_reposink.

9.52 nnstreamer/elements/gsttensor_reposrc.c File Reference

GStreamer plugin to handle tensor repository.

```
#include <string.h>
#include "gsttensor_repo.h"
#include "gsttensor_reposrc.h"
```

Include dependency graph for gsttensor_reposrc.c:



Macros

- #define GST_CAT_DEFAULT gst_tensor_reposrc_debug
- #define CAPS_STRING GST_TENSOR_CAP_DEFAULT "; " GST_TENSORS_CAP_DEFAULT
- #define DEFAULT_SILENT TRUE
- #define DEFAULT_INDEX 0
- #define INVALID_INDEX G_MAXUINT
- #define gst_tensor_reposrc_parent_class parent_class

Enumerations

- enum { PROP_0, PROP_CAPS, PROP_SLOT_ID, PROP_SILENT }

  tensor_reposrc properties

Functions

- GST_DEBUG_CATEGORY_STATIC (gst_tensor_reposrc_debug)
- static void gst_tensor_reposrc_set_property (GObject *object, guint prop_id, const GValue *value, GParamSpec *pspec)

  set property of tensor_reposrc
- static void gst_tensor_reposrc_get_property (GObject *object, guint prop_id, GValue *value, GParamSpec *pspec)

  get property of tensor_reposrc
- static void gst_tensor_reposrc_dispose (GObject *object)

  object dispose of tensor_reposrc
- static GstCaps * gst_tensor_reposrc_getcaps (GstBaseSrc *src, GstCaps *filter)

  get cap of tensor_reposrc
- static GstFlowReturn gst_tensor_reposrc_create (GstPushSrc *src, GstBuffer **buffer)

  create func of tensor_reposrc
- G_DEFINE_TYPE (GstTensorRepoSrc, gst_tensor_reposrc, GST_TYPE_PUSH_SRC)
- static void gst_tensor_reposrc_class_init (GstTensorRepoSrcClass *klass)

  class initialization of tensor_reposrc
- static void gst_tensor_reposrc_init (GstTensorRepoSrc *self)

  object initialization of tensor_reposrc
- static GstBuffer * gst_tensor_reposrc_gen_dummy_buffer (GstTensorRepoSrc *self)

  create dummy buffer for initialization

9.52.1 Detailed Description

GStreamer plugin to handle tensor repository.

GStreamer Copyright (C) 2005 Thomas Vander Stichele thomas@apestaart.org Copyright (C) 2005 Ronald S. Bultje rbultje@ronald.bitfreak.net Copyright (C) 2018 Samsung Electronics Co., Ltd.

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Library General Public License as published by the Free Software Foundation; version 2.1 of the License.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Library General Public License for more details. SECTION: element-tensor_reposrc

Pop element to handle tensor repo

Date

19 Nov 2018

See also

https://github.com/nnstreamer/nnstreamer

Author

Jijoong Moon jijoong.moon@samsung.com

Bug No known bugs except for NYI items

9.52.2 Macro Definition Documentation

9.52.2.1 CAPS_STRING

```
#define CAPS_STRING GST_TENSOR_CAP_DEFAULT "; " GST_TENSORS_CAP_DEFAULT
```

Definition at line 41 of file gsttensor_reposrc.c.

9.52.2.2 DEFAULT_INDEX

```
#define DEFAULT_INDEX 0
```

Definition at line 55 of file gsttensor_reposrc.c.

9.52.2.3 DEFAULT_SILENT

```
#define DEFAULT_SILENT TRUE
```

Definition at line 54 of file gsttensor_reposrc.c.

9.52.2.4 GST_CAT_DEFAULT

```
#define GST_CAT_DEFAULT gst_tensor_reposrc_debug
```

Definition at line 40 of file gsttensor_reposrc.c.

9.52.2.5 gst_tensor_reposrc_parent_class

```
#define gst_tensor_reposrc_parent_class parent_class
```

Definition at line 67 of file gsttensor_reposrc.c.

9.52.2.6 INVALID_INDEX

```
#define INVALID_INDEX G_MAXUINT
```

Definition at line 56 of file gsttensor_reposrc.c.

9.52.3 Enumeration Type Documentation

9.52.3.1 anonymous enum

```
anonymous enum
```

tensor_reposrc properties

Enumerator

PROP_0	
PROP_CAPS	
PROP_SLOT_ID	
PROP_SILENT	

Definition at line 46 of file gsttensor_reposrc.c.

9.52.4 Function Documentation

9.52.4.1 G_DEFINE_TYPE()

```
G_DEFINE_TYPE (
            GstTensorRepoSrc ,
            gst_tensor_reposrc ,
            GST_TYPE_PUSH_SRC  )
```

9.52.4.2 GST_DEBUG_CATEGORY_STATIC()

```
GST_DEBUG_CATEGORY_STATIC (
            gst_tensor_reposrc_debug  )
```

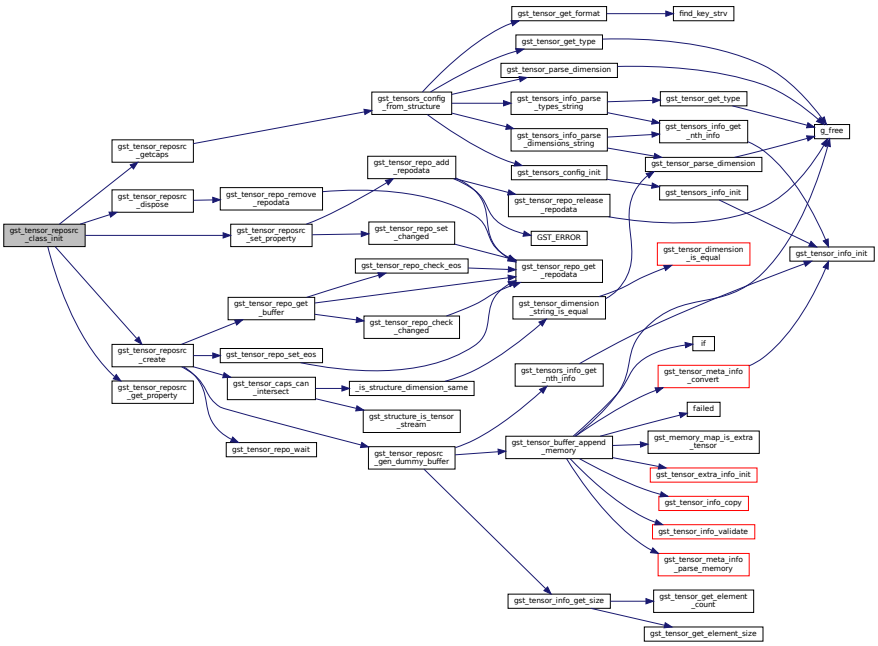
9.52.4.3 gst_tensor_reposrc_class_init()

```
static void gst_tensor_reposrc_class_init (
            GstTensorRepoSrcClass * klass )  [static]
```

class initialization of tensor_reposrc

Definition at line 74 of file gsttensor_reposrc.c.

Here is the call graph for this function:



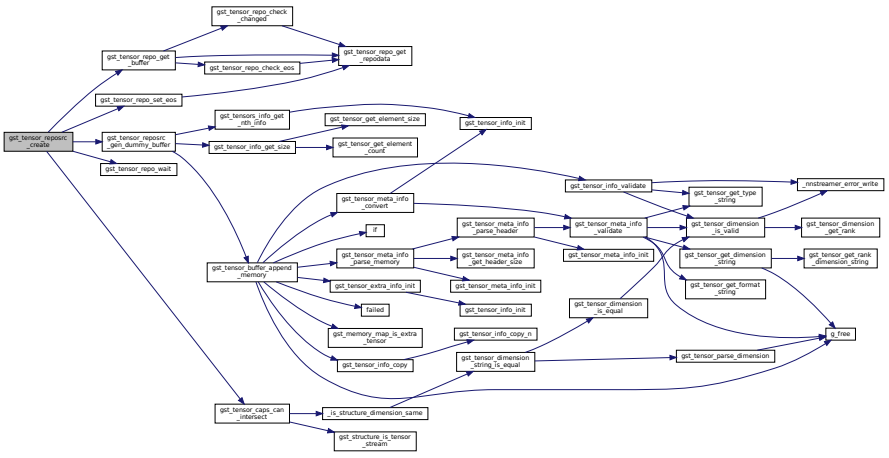
9.52.4.4 gst_tensor_reposrc_create()

```
static GstFlowReturn gst_tensor_reposrc_create (
            GstPushSrc * src,
            GstBuffer ** buffer )  [static]
```

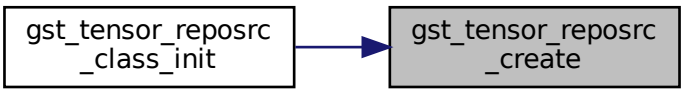
create func of tensor_reposrc

Definition at line 325 of file gsttensor_reposrc.c.

Here is the call graph for this function:



Here is the caller graph for this function:



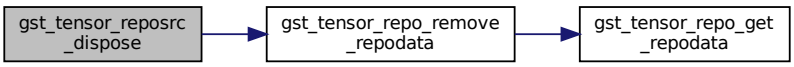
9.52.4.5 gst_tensor_reposrc_dispose()

```
static void gst_tensor_reposrc_dispose (
             GObject * object )  [static]
```

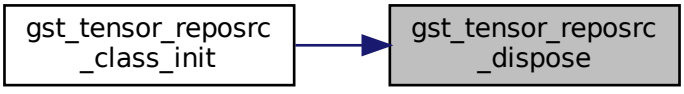
object dispose of tensor_reposrc

Definition at line 141 of file gsttensor_reposrc.c.

Here is the call graph for this function:



Here is the caller graph for this function:



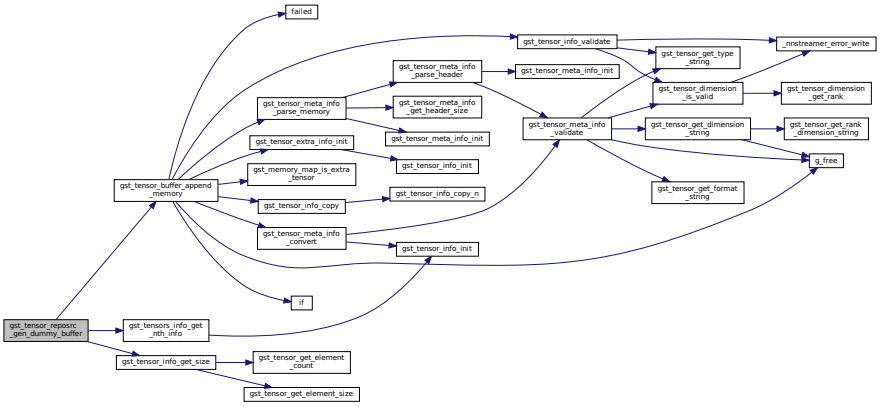
9.52.4.6 gst_tensor_reposrc_gen_dummy_buffer()

```
static GstBuffer* gst_tensor_reposrc_gen_dummy_buffer (
            GstTensorRepoSrc * self )  [static]
```

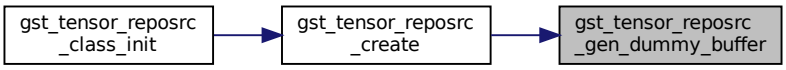
create dummy buffer for initialization

Definition at line 290 of file gsttensor_reposrc.c.

Here is the call graph for this function:



Here is the caller graph for this function:



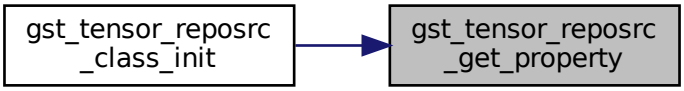
9.52.4.7 gst_tensor_reposrc_get_property()

```
static void gst_tensor_reposrc_get_property (
            GObject * object,
            guint prop_id,
            GValue * value,
            GParamSpec * pspec )  [static]
```

get property of tensor_reposrc

Definition at line 265 of file gsttensor_reposrc.c.

Here is the caller graph for this function:



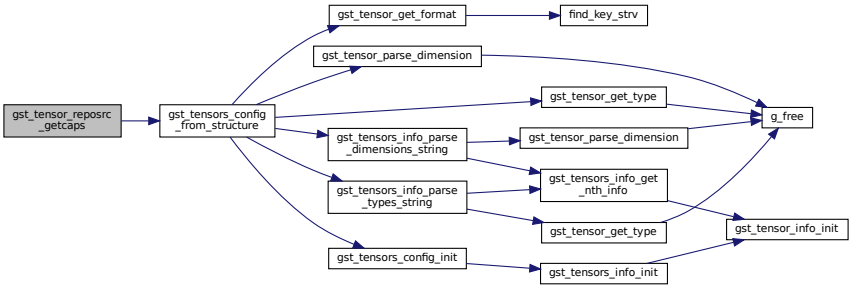
9.52.4.8 gst_tensor_reposrc_getcaps()

```
static GstCaps * gst_tensor_reposrc_getcaps (
            GstBaseSrc * src,
            GstCaps * filter )  [static]
```

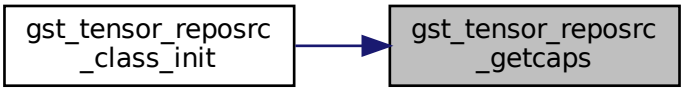
get cap of tensor_reposrc

Definition at line 160 of file gsttensor_reposrc.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.52.4.9 gst_tensor_reposrc_init()

```
static void gst_tensor_reposrc_init (
            GstTensorRepoSrc * self )  [static]
```

object initialization of tensor_reposrc

Definition at line 126 of file gsttensor_reposrc.c.

Here is the call graph for this function:

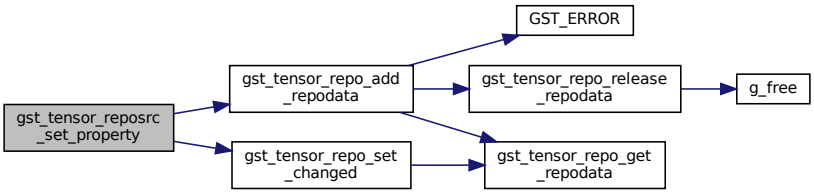
9.52.4.10 gst_tensor_reposrc_set_property()

```
static void gst_tensor_reposrc_set_property (
            GObject * object,
            guint prop_id,
            const GValue * value,
            GParamSpec * pspec )  [static]
```

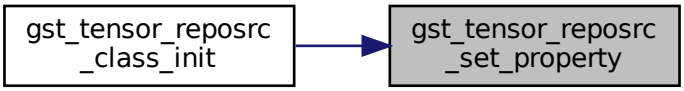
set property of tensor_reposrc

Definition at line 201 of file gsttensor_reposrc.c.

Here is the call graph for this function:



Here is the caller graph for this function:

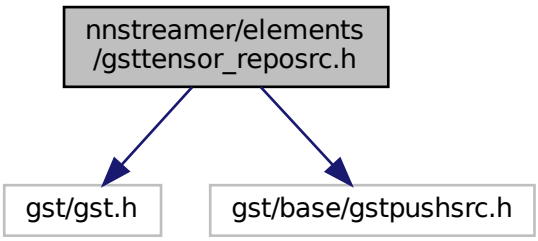


9.53 nnstreamer/elements/gsttensor_reposrc.h File Reference

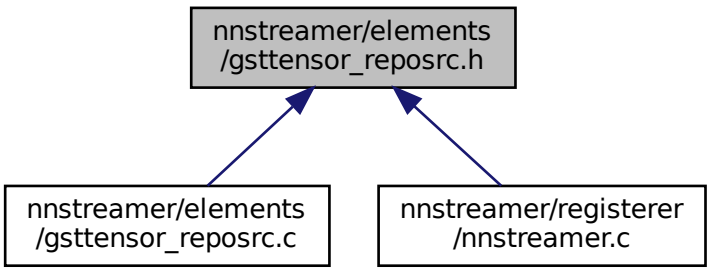
GStreamer plugin to handle tensor repository.

```
#include <gst/gst.h>
#include <gst/base/gstpushsrc.h>
```

Include dependency graph for gsttensor_reposrc.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct _GstTensorRepoSrc

  GstTensorRepoSrc data structure.
- struct _GstTensorRepoSrcClass

  GstTensorRepoSrcClass data structure.

Macros

- #define GST_TYPE_TENSOR_REPOSRC (gst_tensor_reposrc_get_type())
- #define GST_TENSOR_REPOSRC(obj) (G_TYPE_CHECK_INSTANCE_CAST((obj),GST_TYPE_TENSOR_REPOSRC,GstTe
- #define GST_TENSOR_REPOSRC_CLASS(klass) (G_TYPE_CHECK_CLASS_CAST((klass),GST_TYPE_TENSOR_REPOS
- #define GST_IS_TENSOR_REPOSRC(obj) (G_TYPE_CHECK_INSTANCE_TYPE((obj),GST_TYPE_TENSOR_REPOSRC))
- #define GST_IS_TENSOR_REPOSRC_CLASS(klass) (G_TYPE_CHECK_CLASS_TYPE((klass),GST_TYPE_TENSOR_REP

Typedefs

- typedef struct _GstTensorRepoSrc GstTensorRepoSrc
- typedef struct _GstTensorRepoSrcClass GstTensorRepoSrcClass

Functions

- GType gst_tensor_reposrc_get_type (void)

  Function to get type of tensor_reposrc.

9.53.1 Detailed Description

GStreamer plugin to handle tensor repository.

GStreamer Copyright (C) 2005 Thomas Vander Stichele thomas@apestaart.org Copyright (C) 2005 Ronald S. Bultje rbultje@ronald.bitfreak.net Copyright (C) 2018 Samsung Electronics Co., Ltd.

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Library General Public License as published by the Free Software Foundation; version 2.1 of the License.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Library General Public License for more details.

Date

19 Nov 2018

See also

https://github.com/nnstreamer/nnstreamer

Author

Jijoong Moon jijoong.moon@samsung.com

Bug No known bugs except for NYI items

9.53.2 Macro Definition Documentation

9.53.2.1 GST_IS_TENSOR_REPOSRC

```
#define GST_IS_TENSOR_REPOSRC(
              obj ) (G_TYPE_CHECK_INSTANCE_TYPE((obj),GST_TYPE_TENSOR_REPOSRC))
```

Definition at line 41 of file gsttensor_reposrc.h.

9.53.2.2 GST_IS_TENSOR_REPOSRC_CLASS

```
#define GST_IS_TENSOR_REPOSRC_CLASS(
              klass ) (G_TYPE_CHECK_CLASS_TYPE((klass),GST_TYPE_TENSOR_REPOSRC))
```

Definition at line 43 of file gsttensor_reposrc.h.

9.53.2.3 GST_TENSOR_REPOSRC

```
#define GST_TENSOR_REPOSRC(
              obj ) (G_TYPE_CHECK_INSTANCE_CAST((obj),GST_TYPE_TENSOR_REPOSRC,GstTensorRepoSrc))
```

Definition at line 37 of file gsttensor_reposrc.h.

9.53.2.4 GST_TENSOR_REPOSRC_CLASS

```
#define GST_TENSOR_REPOSRC_CLASS(
              klass ) (G_TYPE_CHECK_CLASS_CAST((klass),GST_TYPE_TENSOR_REPOSRC,GstTensorRepoSrcClass))
```

Definition at line 39 of file gsttensor_reposrc.h.

9.53.2.5 GST_TYPE_TENSOR_REPOSRC

```
#define GST_TYPE_TENSOR_REPOSRC (gst_tensor_reposrc_get_type())
```

Definition at line 35 of file gsttensor_reposrc.h.

9.53.3 Typedef Documentation

9.53.3.1 GstTensorRepoSrc

```
typedef struct _GstTensorRepoSrc GstTensorRepoSrc
```

Definition at line 46 of file gsttensor_reposrc.h.

9.53.3.2 GstTensorRepoSrcClass

```
typedef struct _GstTensorRepoSrcClass GstTensorRepoSrcClass
```

Definition at line 47 of file gsttensor_reposrc.h.

9.53.4 Function Documentation

9.53.4.1 gst_tensor_reposrc_get_type()

```
GType gst_tensor_reposrc_get_type (
            void  )
```

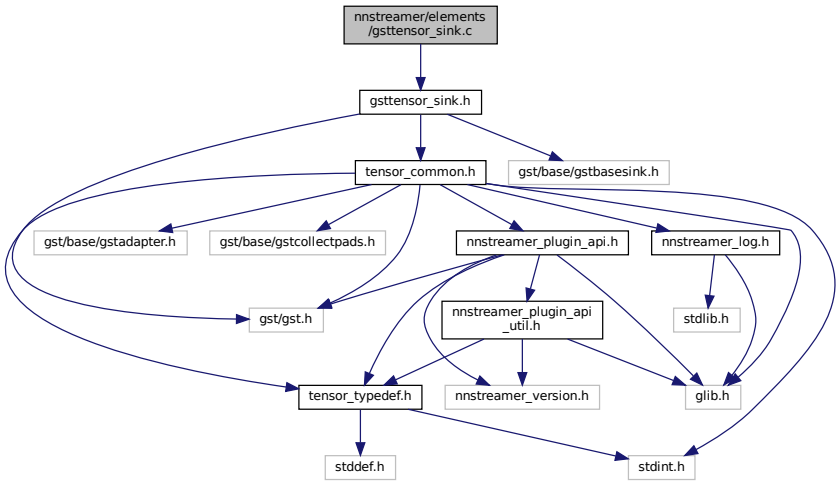
Function to get type of tensor_reposrc.

9.54 nnstreamer/elements/gsttensor_sink.c File Reference

GStreamer plugin to handle tensor stream.

```
#include "gsttensor_sink.h"
```

Include dependency graph for gsttensor_sink.c:



Macros

- #define DBG (!self->silent)

  Macro for debug mode.
- #define silent_debug_timestamp(self, buf)
- #define GST_CAT_DEFAULT gst_tensor_sink_debug
- #define DEFAULT_EMIT_SIGNAL TRUE

  Flag to emit signals.
- #define DEFAULT_SIGNAL_RATE 0

  New data signals per second.
- #define DEFAULT_SILENT TRUE

  Flag to print minimized log.
- #define DEFAULT_QOS TRUE

  Flag for qos event.
- #define DEFAULT_SYNC FALSE

  Flag to synchronize on the clock (Default FALSE). It may be delayed with tensor_filter element, to invoke neural network model. See GstBaseSink::sync property for more details.
- #define gst_tensor_sink_parent_class parent_class

Enumerations

- enum { SIGNAL_NEW_DATA, SIGNAL_STREAM_START, SIGNAL_EOS, LAST_SIGNAL }

  tensor_sink signals.
- enum { PROP_0, PROP_SIGNAL_RATE, PROP_EMIT_SIGNAL, PROP_SILENT }

  tensor_sink properties.

Functions

- GST_DEBUG_CATEGORY_STATIC (gst_tensor_sink_debug)
- static void gst_tensor_sink_set_property (GObject *object, guint prop_id, const GValue *value, GParamSpec *pspec)

  Setter for tensor_sink properties.
- static void gst_tensor_sink_get_property (GObject *object, guint prop_id, GValue *value, GParamSpec *pspec)

  Getter for tensor_sink properties.
- static void gst_tensor_sink_finalize (GObject *object)

  Function to finalize instance.
- static gboolean gst_tensor_sink_event (GstBaseSink *sink, GstEvent *event)

  Handle events.
- static gboolean gst_tensor_sink_query (GstBaseSink *sink, GstQuery *query)

  Handle queries.
- static GstFlowReturn gst_tensor_sink_render (GstBaseSink *sink, GstBuffer *buffer)

  Handle buffer.
- static GstFlowReturn gst_tensor_sink_render_list (GstBaseSink *sink, GstBufferList *buffer_list)

  Handle list of buffers.
- static void gst_tensor_sink_render_buffer (GstTensorSink *self, GstBuffer *buffer)

  Handle buffer data.
- static void gst_tensor_sink_set_last_render_time (GstTensorSink *self, GstClockTime now)

  Setter for value last_render_time.
- static GstClockTime gst_tensor_sink_get_last_render_time (GstTensorSink *self)

Getter for value last_render_time.

- static void gst_tensor_sink_set_signal_rate (GstTensorSink *self, guint rate)

  Setter for value signal_rate.
- static guint gst_tensor_sink_get_signal_rate (GstTensorSink *self)

  Getter for value signal_rate.
- static void gst_tensor_sink_set_emit_signal (GstTensorSink *self, gboolean emit)

  Setter for flag emit_signal.
- static gboolean gst_tensor_sink_get_emit_signal (GstTensorSink *self)

  Getter for flag emit_signal.
- static void gst_tensor_sink_set_silent (GstTensorSink *self, gboolean silent)

  Setter for flag silent.
- static gboolean gst_tensor_sink_get_silent (GstTensorSink *self)

  Getter for flag silent.
- G_DEFINE_TYPE (GstTensorSink, gst_tensor_sink, GST_TYPE_BASE_SINK)
- static void gst_tensor_sink_class_init (GstTensorSinkClass *klass)

  Initialize tensor_sink class.
- static void gst_tensor_sink_init (GstTensorSink *self)

  Initialize tensor_sink element.

Variables

- static guint _tensor_sink_signals [LAST_SIGNAL] = { 0 }

  Variable for signal ids.

9.54.1 Detailed Description

GStreamer plugin to handle tensor stream.

GStreamer Copyright (C) 2005 Thomas Vander Stichele thomas@apestaart.org Copyright (C) 2005 Ronald S. Bultje rbultje@ronald.bitfreak.net Copyright (C) 2018 Samsung Electronics Co., Ltd.

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Library General Public License as published by the Free Software Foundation; version 2.1 of the License.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Library General Public License for more details. SECTION:element-tensor_sink

Sink element to handle tensor stream

Date

15 June 2018

See also

https://github.com/nnstreamer/nnstreamer

Author

Jaeyun Jung jy1210.jung@samsung.com

Bug No known bugs except for NYI items

9.54.2 Macro Definition Documentation

9.54.2.1 DBG

```
#define DBG (!self->silent)
```

Macro for debug mode.

Definition at line 41 of file gsttensor_sink.c.

9.54.2.2 DEFAULT_EMIT_SIGNAL

```
#define DEFAULT_EMIT_SIGNAL TRUE
```

Flag to emit signals.

Definition at line 80 of file gsttensor_sink.c.

9.54.2.3 DEFAULT_QOS

```
#define DEFAULT_QOS TRUE
```

Flag for qos event.

See GstBaseSink::qos property for more details.

Definition at line 97 of file gsttensor_sink.c.

9.54.2.4 DEFAULT_SIGNAL_RATE

```
#define DEFAULT_SIGNAL_RATE 0
```

New data signals per second.

Definition at line 85 of file gsttensor_sink.c.

9.54.2.5 DEFAULT_SILENT

```
#define DEFAULT_SILENT TRUE
```

Flag to print minimized log.

Definition at line 90 of file gsttensor_sink.c.

9.54.2.6 DEFAULT_SYNC

```
#define DEFAULT_SYNC FALSE
```

Flag to synchronize on the clock (Default FALSE). It may be delayed with tensor_filter element, to invoke neural network model. See GstBaseSink::sync property for more details.

Definition at line 104 of file gsttensor_sink.c.

9.54.2.7 GST_CAT_DEFAULT

```
#define GST_CAT_DEFAULT gst_tensor_sink_debug
```

Definition at line 53 of file gsttensor_sink.c.

9.54.2.8 gst_tensor_sink_parent_class

```
#define gst_tensor_sink_parent_class parent_class
```

Definition at line 140 of file gsttensor_sink.c.

9.54.2.9 silent_debug_timestamp

```
#define silent_debug_timestamp(
              self,
              buf )
```

Value:

```
  do { \
  if (DBG) { \
    GST_DEBUG_OBJECT (self, "pts = %" GST_TIME_FORMAT, GST_TIME_ARGS (GST_BUFFER_PTS (buf))); \
    GST_DEBUG_OBJECT (self, "dts = %" GST_TIME_FORMAT, GST_TIME_ARGS (GST_BUFFER_DTS (buf))); \
    GST_DEBUG_OBJECT (self, "duration = %" GST_TIME_FORMAT "\n", GST_TIME_ARGS (GST_BUFFER_DURATION (buf)));
       \
  } \
} while (0)
```

Definition at line 44 of file gsttensor_sink.c.

9.54.3 Enumeration Type Documentation

9.54.3.1 anonymous enum

```
anonymous enum
```

tensor_sink signals.

Enumerator

SIGNAL_NEW_DATA	
SIGNAL_STREAM_START	
SIGNAL_EOS	
LAST_SIGNAL	

Definition at line 58 of file gsttensor_sink.c.

9.54.3.2 anonymous enum

```
anonymous enum
```

tensor_sink properties.

Enumerator

PROP_0	
PROP_SIGNAL_RATE	
PROP_EMIT_SIGNAL	
PROP_SILENT	

Definition at line 69 of file gsttensor_sink.c.

9.54.4 Function Documentation

9.54.4.1 G_DEFINE_TYPE()

```
G_DEFINE_TYPE (
              GstTensorSink ,
              gst_tensor_sink ,
              GST_TYPE_BASE_SINK  )
```

9.54.4.2 GST_DEBUG_CATEGORY_STATIC()

```
GST_DEBUG_CATEGORY_STATIC (
              gst_tensor_sink_debug  )
```

9.54.4.3 gst_tensor_sink_class_init()

```
static void gst_tensor_sink_class_init (
            GstTensorSinkClass * klass )  [static]
```

Initialize tensor_sink class.

GObject methods

GstTensorSink::signal-rate:

The number of new data signals per second (Default 0 for unlimited, MAX 500) If signal-rate is larger than 0, GstTensorSink calculates the time to emit a signal with this property. If set 0 (default value), all the received buffers will be passed to the application.

Please note that this property does not guarantee the periodic signals. This means if GstTensorSink cannot get the buffers in time, it will pass all the buffers. (working like default 0)

GstTensorSink::emit-signal:

The flag to emit the signals for new data, stream start, and eos.

GstTensorSink::silent:

The flag to enable/disable debugging messages.

GstTensorSink::new-data:

Signal to get the buffer from GstTensorSink.

GstTensorSink::stream-start:

Signal to indicate the start of a new stream. Optional. An application can use this signal to detect the start of a new stream, instead of the message GST_MESSAGE_STREAM_START from pipeline.

GstTensorSink::eos:

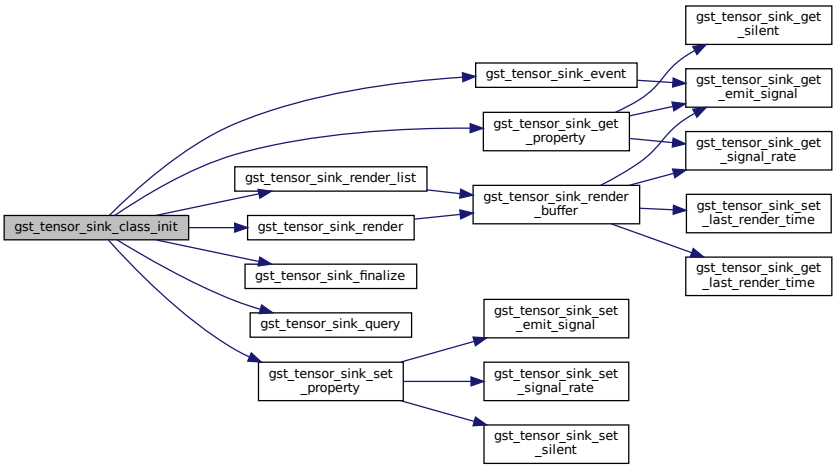
Signal to indicate the end-of-stream. Optional. An application can use this signal to detect the EOS (end-of-stream), instead of the message GST_MESSAGE_EOS from pipeline.

pad template

GstBaseSink methods

Definition at line 147 of file gsttensor_sink.c.

Here is the call graph for this function:



9.54.4.4 gst_tensor_sink_event()

```
static gboolean gst_tensor_sink_event (
            GstBaseSink * sink,
            GstEvent * event )  [static]
```

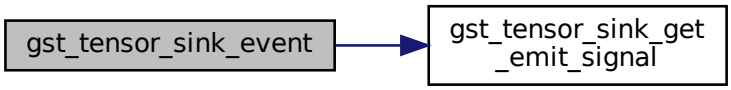
Handle events.

GstBaseSink method implementation

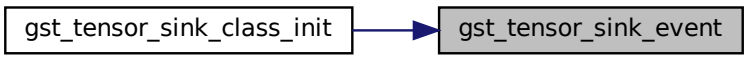
GstBaseSink method implementation.

Definition at line 365 of file gsttensor_sink.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.54.4.5 gst_tensor_sink_finalize()

```
static void gst_tensor_sink_finalize (
            GObject * object )  [static]
```

Function to finalize instance.

GObject method implementation.

Definition at line 348 of file gsttensor_sink.c.

Here is the caller graph for this function:

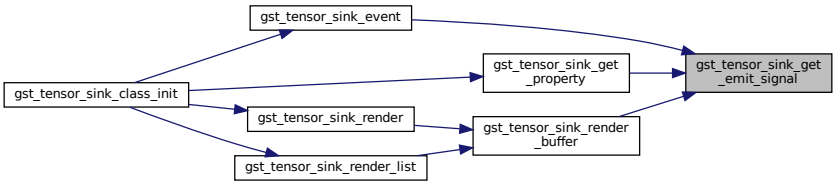
9.54.4.6 gst_tensor_sink_get_emit_signal()

```
static gboolean gst_tensor_sink_get_emit_signal (
            GstTensorSink * self )  [static]
```

Getter for flag emit_signal.

Definition at line 610 of file gsttensor_sink.c.

Here is the caller graph for this function:



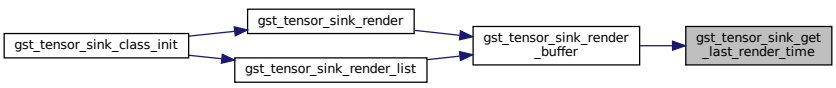
9.54.4.7 gst_tensor_sink_get_last_render_time()

```
static GstClockTime gst_tensor_sink_get_last_render_time (
            GstTensorSink * self )  [static]
```

Getter for value last_render_time.

Definition at line 548 of file gsttensor_sink.c.

Here is the caller graph for this function:



9.54.4.8 gst_tensor_sink_get_property()

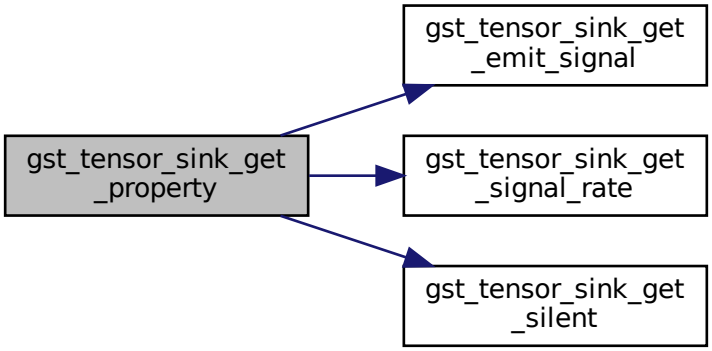
```
static void gst_tensor_sink_get_property (
             GObject * object,
             guint prop_id,
             GValue * value,
             GParamSpec * pspec )  [static]
```

Getter for tensor_sink properties.

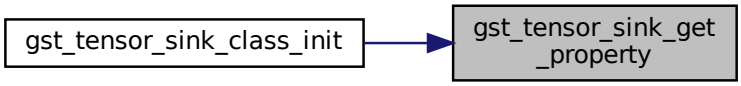
GObject method implementation.

Definition at line 316 of file gsttensor_sink.c.

Here is the call graph for this function:



Here is the caller graph for this function:



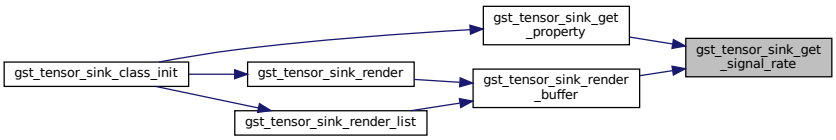
9.54.4.9 gst_tensor_sink_get_signal_rate()

```
static guint gst_tensor_sink_get_signal_rate (
            GstTensorSink * self )  [static]
```

Getter for value signal_rate.

Definition at line 579 of file gsttensor_sink.c.

Here is the caller graph for this function:



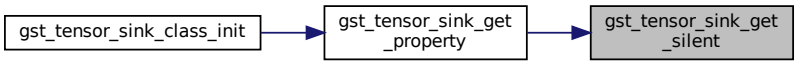
9.54.4.10 gst_tensor_sink_get_silent()

```
static gboolean gst_tensor_sink_get_silent (
            GstTensorSink * self )  [static]
```

Getter for flag silent.

Definition at line 639 of file gsttensor_sink.c.

Here is the caller graph for this function:



9.54.4.11 gst_tensor_sink_init()

```
static void gst_tensor_sink_init (
            GstTensorSink * self )  [static]
```

Initialize tensor_sink element.

init properties

enable qos

Definition at line 257 of file gsttensor_sink.c.

9.54.4.12 gst_tensor_sink_query()

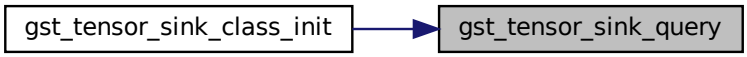
```
static gboolean gst_tensor_sink_query (
            GstBaseSink * sink,
            GstQuery * query )  [static]
```

Handle queries.

GstBaseSink method implementation. tensor sink does not support seeking

Definition at line 406 of file gsttensor_sink.c.

Here is the caller graph for this function:



9.54.4.13 gst_tensor_sink_render()

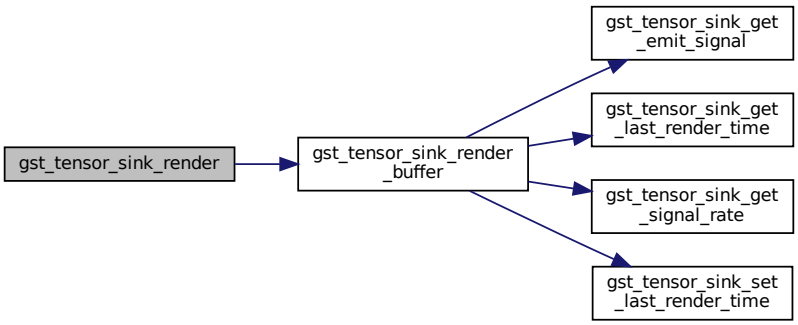
```
static GstFlowReturn gst_tensor_sink_render (
            GstBaseSink * sink,
            GstBuffer * buffer )  [static]
```

Handle buffer.

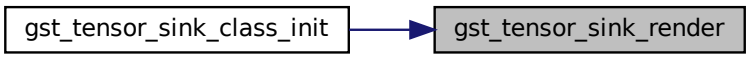
GstBaseSink method implementation.

Definition at line 438 of file gsttensor_sink.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.54.4.14 gst_tensor_sink_render_buffer()

```
static void gst_tensor_sink_render_buffer (
             GstTensorSink * self,
             GstBuffer * buffer )  [static]
```

Handle buffer data.

internal functions

Returns

None

Parameters

self	pointer to GstTensorSink
buffer	pointer to GstBuffer to be handled

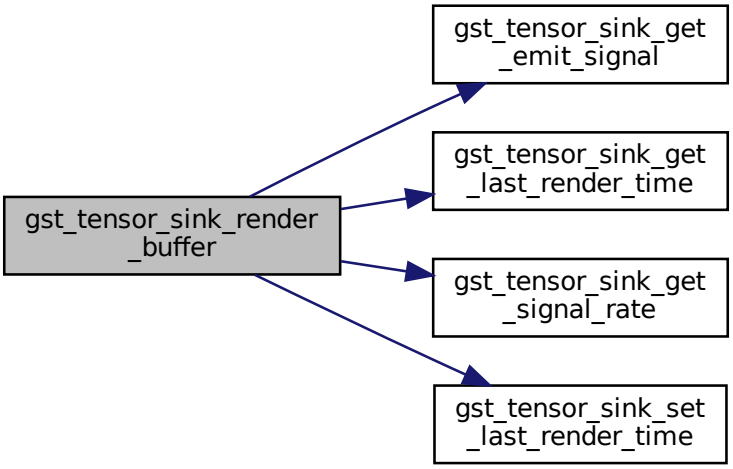
time for next signal

send data after render time, or firstly received buffer

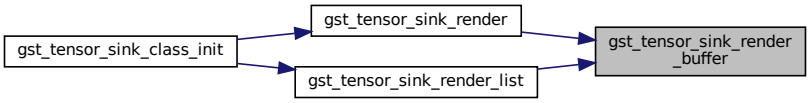
send data if signal rate is 0

Definition at line 479 of file gsttensor_sink.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.54.4.15 gst_tensor_sink_render_list()

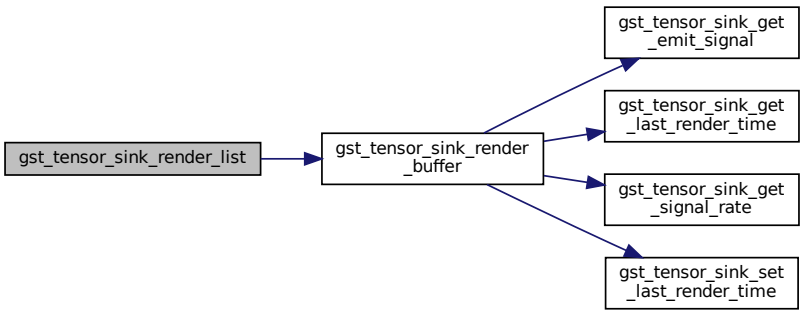
```
static GstFlowReturn gst_tensor_sink_render_list (
            GstBaseSink * sink,
            GstBufferList * buffer_list )  [static]
```

Handle list of buffers.

GstBaseSink method implementation.

Definition at line 454 of file gsttensor_sink.c.

Here is the call graph for this function:



Here is the caller graph for this function:

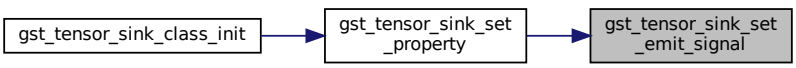
9.54.4.16 gst_tensor_sink_set_emit_signal()

```
static void gst_tensor_sink_set_emit_signal (
            GstTensorSink * self,
            gboolean emit )  [static]
```

Setter for flag emit_signal.

Definition at line 596 of file gsttensor_sink.c.

Here is the caller graph for this function:



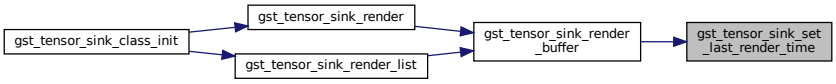
9.54.4.17 gst_tensor_sink_set_last_render_time()

```
static void gst_tensor_sink_set_last_render_time (
            GstTensorSink * self,
            GstClockTime now )  [static]
```

Setter for value last_render_time.

Definition at line 535 of file gsttensor_sink.c.

Here is the caller graph for this function:



9.54.4.18 gst_tensor_sink_set_property()

```
static void gst_tensor_sink_set_property (
            GObject * object,
            guint prop_id,
            const GValue * value,
            GParamSpec * pspec )  [static]
```

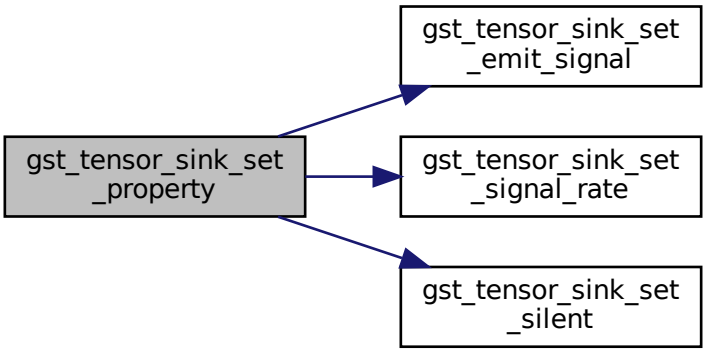
Setter for tensor_sink properties.

GObject method implementation

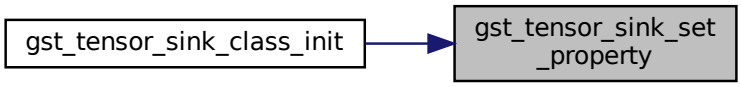
GObject method implementation.

Definition at line 284 of file gsttensor_sink.c.

Here is the call graph for this function:



Here is the caller graph for this function:



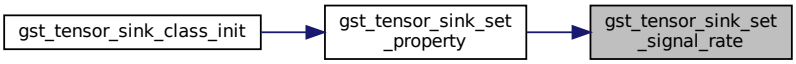
9.54.4.19 gst_tensor_sink_set_signal_rate()

```
static void gst_tensor_sink_set_signal_rate (
            GstTensorSink * self,
            guint rate )  [static]
```

Setter for value signal_rate.

Definition at line 565 of file gsttensor_sink.c.

Here is the caller graph for this function:



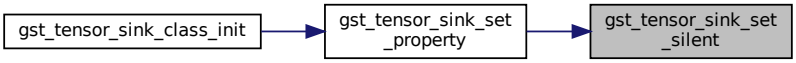
9.54.4.20 gst_tensor_sink_set_silent()

```
static void gst_tensor_sink_set_silent (
            GstTensorSink * self,
            gboolean silent )  [static]
```

Setter for flag silent.

Definition at line 627 of file gsttensor_sink.c.

Here is the caller graph for this function:



9.54.5 Variable Documentation

9.54.5.1 _tensor_sink_signals

```
guint _tensor_sink_signals[LAST_SIGNAL] = { 0 }  [static]
```

Variable for signal ids.

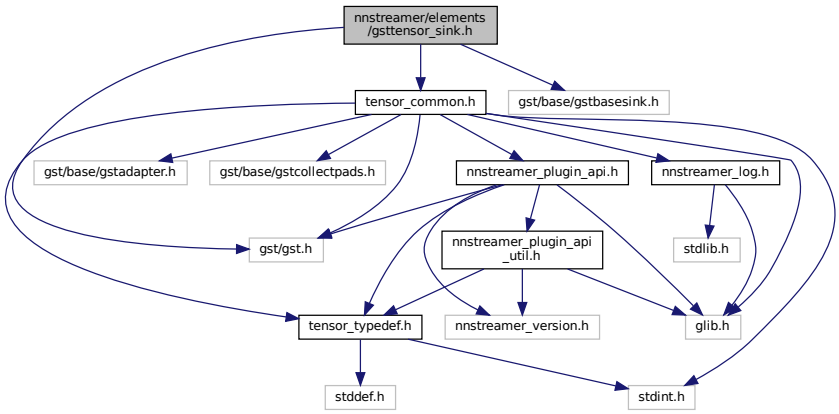
Definition at line 109 of file gsttensor_sink.c.

9.55 nnstreamer/elements/gsttensor_sink.h File Reference

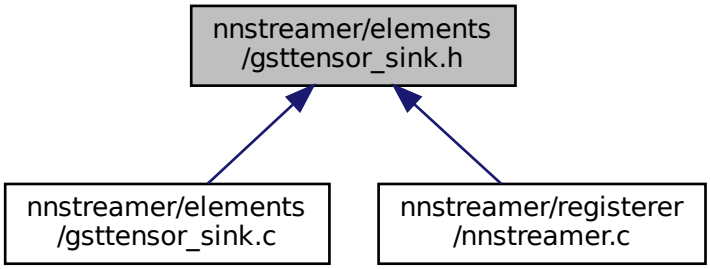
GStreamer plugin to handle tensor stream.

```
#include <gst/gst.h>
#include <gst/base/gstbasesink.h>
#include <tensor_common.h>
```

Include dependency graph for gsttensor_sink.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct _GstTensorSink

  GstTensorSink data structure.
- struct _GstTensorSinkClass

  GstTensorSinkClass data structure.

Macros

- #define GST_TYPE_TENSOR_SINK (gst_tensor_sink_get_type())
- #define GST_TENSOR_SINK(obj) (G_TYPE_CHECK_INSTANCE_CAST((obj),GST_TYPE_TENSOR_SINK,GstTensorSink))
- #define GST_TENSOR_SINK_CLASS(klass) (G_TYPE_CHECK_CLASS_CAST((klass),GST_TYPE_TENSOR_SINK,GstTens
- #define GST_IS_TENSOR_SINK(obj) (G_TYPE_CHECK_INSTANCE_TYPE((obj),GST_TYPE_TENSOR_SINK))
- #define GST_IS_TENSOR_SINK_CLASS(klass) (G_TYPE_CHECK_CLASS_TYPE((klass),GST_TYPE_TENSOR_SINK))

Typedefs

- typedef struct _GstTensorSink GstTensorSink
- typedef struct _GstTensorSinkClass GstTensorSinkClass

Functions

- GType gst_tensor_sink_get_type (void)

  Function to get type of tensor_sink.

9.55.1 Detailed Description

GStreamer plugin to handle tensor stream.

GStreamer Copyright (C) 2005 Thomas Vander Stichele thomas@apestaart.org Copyright (C) 2005 Ronald S. Bultje rbultje@ronald.bitfreak.net Copyright (C) 2018 Samsung Electronics Co., Ltd.

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Library General Public License as published by the Free Software Foundation; version 2.1 of the License.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Library General Public License for more details.

Date

15 June 2018

See also

https://github.com/nnstreamer/nnstreamer

Author

Jaeyun Jung jy1210.jung@samsung.com

Bug No known bugs except for NYI items

9.55.2 Macro Definition Documentation

9.55.2.1 GST_IS_TENSOR_SINK

```
#define GST_IS_TENSOR_SINK(
              obj ) (G_TYPE_CHECK_INSTANCE_TYPE((obj),GST_TYPE_TENSOR_SINK))
```

Definition at line 42 of file gsttensor_sink.h.

9.55.2.2 GST_IS_TENSOR_SINK_CLASS

```
#define GST_IS_TENSOR_SINK_CLASS(
              klass ) (G_TYPE_CHECK_CLASS_TYPE((klass),GST_TYPE_TENSOR_SINK))
```

Definition at line 44 of file gsttensor_sink.h.

9.55.2.3 GST_TENSOR_SINK

```
#define GST_TENSOR_SINK(
              obj ) (G_TYPE_CHECK_INSTANCE_CAST((obj),GST_TYPE_TENSOR_SINK,GstTensorSink))
```

Definition at line 38 of file gsttensor_sink.h.

9.55.2.4 GST_TENSOR_SINK_CLASS

```
#define GST_TENSOR_SINK_CLASS(
              klass ) (G_TYPE_CHECK_CLASS_CAST((klass),GST_TYPE_TENSOR_SINK,GstTensorSinkClass))
```

Definition at line 40 of file gsttensor_sink.h.

9.55.2.5 GST_TYPE_TENSOR_SINK

```
#define GST_TYPE_TENSOR_SINK (gst_tensor_sink_get_type())
```

Definition at line 36 of file gsttensor_sink.h.

9.55.3 Typedef Documentation

9.55.3.1 GstTensorSink

```
typedef struct _GstTensorSink GstTensorSink
```

Definition at line 47 of file gsttensor_sink.h.

9.55.3.2 GstTensorSinkClass

```
typedef struct _GstTensorSinkClass GstTensorSinkClass
```

Definition at line 48 of file gsttensor_sink.h.

9.55.4 Function Documentation

9.55.4.1 gst_tensor_sink_get_type()

```
GType gst_tensor_sink_get_type (
            void  )
```

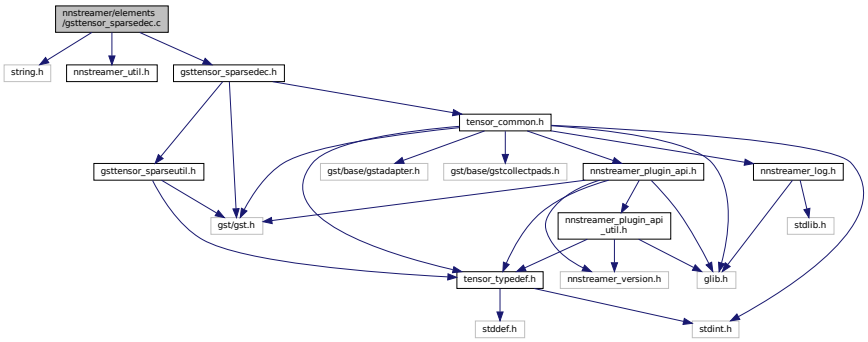
Function to get type of tensor_sink.

9.56 nnstreamer/elements/gsttensor_sparsedec.c File Reference

GStreamer element to decode sparse tensors into dense tensors.

```
#include <string.h>
#include <nnstreamer_util.h>
#include "gsttensor_sparsedec.h"
```

Include dependency graph for gsttensor_sparsedec.c:



Macros

- #define DBG (!self->silent)

  Macro for debug mode.
- #define GST_CAT_DEFAULT gst_tensor_sparse_dec_debug
- #define DEFAULT_SILENT TRUE

  Flag to print minimized log.
- #define gst_tensor_sparse_dec_parent_class parent_class

Enumerations

- enum { PROP_0, PROP_SILENT }

  tensor_sparse_dec properties

Functions

- GST_DEBUG_CATEGORY_STATIC (gst_tensor_sparse_dec_debug)
- G_DEFINE_TYPE (GstTensorSparseDec, gst_tensor_sparse_dec, GST_TYPE_ELEMENT)
- static void gst_tensor_sparse_dec_finalize (GObject *object)

  Function to finalize instance.
- static void gst_tensor_sparse_dec_set_property (GObject *object, guint prop_id, const GValue *value, GParamSpec *pspec)

  Setter for tensor_sparse_dec properties.
- static void gst_tensor_sparse_dec_get_property (GObject *object, guint prop_id, GValue *value, GParamSpec *pspec)

  Getter for tensor_sparse_dec properties.
- static GstFlowReturn gst_tensor_sparse_dec_chain (GstPad *pad, GstObject *parent, GstBuffer *buf)

  Internal function to transform the input buffer.
- static gboolean gst_tensor_sparse_dec_sink_event (GstPad *pad, GstObject *parent, GstEvent *event)

  This function handles sink pad event.
- static gboolean gst_tensor_sparse_dec_sink_query (GstPad *pad, GstObject *parent, GstQuery *query)

  This function handles sink pad query.
- static void gst_tensor_sparse_dec_class_init (GstTensorSparseDecClass *klass)

  Initialize the tensor_sparse's class.
- static void gst_tensor_sparse_dec_init (GstTensorSparseDec *self)

  Initialize tensor_sparse_dec element.
- static GstCaps * gst_tensor_sparse_dec_query_caps (GstTensorSparseDec *self, GstPad *pad, GstCaps *filter)

  Get pad caps for caps negotiation.

Variables

- static GstStaticPadTemplate sink_template

  Template for sink pad.
- static GstStaticPadTemplate src_template

  Template for src pad.

9.56.1 Detailed Description

GStreamer element to decode sparse tensors into dense tensors.

Date

27 Jul 2021

See also

https://github.com/nnstreamer/nnstreamer

Author

Yongjoo Ahn yongjoo1.ahn@samsung.com

Bug No known bugs except for NYI items

9.56.2 Macro Definition Documentation

9.56.2.1 DBG

```
#define DBG (!self->silent)
```

Macro for debug mode.

SECTION:element-tensor_sparse_dec

tensor_sparse_dec is a GStreamer element to decode incoming sparse tensor into static (dense) format.

The input is always in the format of other/tensors,format=sparse. The output is always in the format of ohter/tensors,format=static.

Please see also tensor_sparse_enc.

<refsect2> <title>Example launch line</title> |[gst-launch-1.0 ... ! other/tensors,format=static ! \ tensor_↩ sparse_enc ! other/tensors,format=sparse ! \ tensor_sparse_dec ! tensor_sink]| </refsect2>

Definition at line 45 of file gsttensor_sparsedec.c.

9.56.2.2 DEFAULT_SILENT

```
#define DEFAULT_SILENT TRUE
```

Flag to print minimized log.

Definition at line 63 of file gsttensor_sparsedec.c.

9.56.2.3 GST_CAT_DEFAULT

```
#define GST_CAT_DEFAULT gst_tensor_sparse_dec_debug
```

Definition at line 49 of file gsttensor_sparsedec.c.

9.56.2.4 gst_tensor_sparse_dec_parent_class

```
#define gst_tensor_sparse_dec_parent_class parent_class
```

Definition at line 81 of file gsttensor_sparsedec.c.

9.56.3 Enumeration Type Documentation

9.56.3.1 anonymous enum

```
anonymous enum
```

tensor_sparse_dec properties

Enumerator

PROP_0	
PROP_SILENT	

Definition at line 54 of file gsttensor_sparsedec.c.

9.56.4 Function Documentation

9.56.4.1 G_DEFINE_TYPE()

```
G_DEFINE_TYPE (
            GstTensorSparseDec ,
            gst_tensor_sparse_dec ,
            GST_TYPE_ELEMENT  )
```

9.56.4.2 GST_DEBUG_CATEGORY_STATIC()

```
GST_DEBUG_CATEGORY_STATIC (
            gst_tensor_sparse_dec_debug  )
```

9.56.4.3 gst_tensor_sparse_dec_chain()

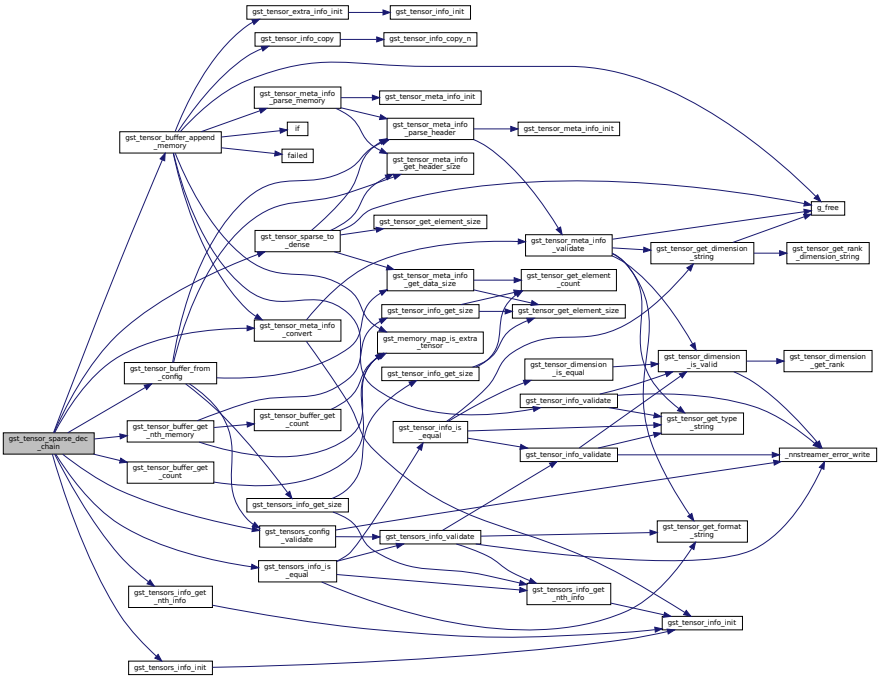
```
static GstFlowReturn gst_tensor_sparse_dec_chain (
            GstPad * pad,
            GstObject * parent,
            GstBuffer * buf )  [static]
```

Internal function to transform the input buffer.

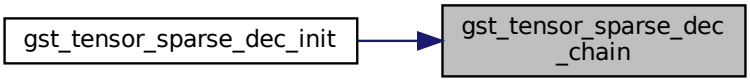
Todo consider more error handling

Definition at line 359 of file gsttensor_sparsedec.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.56.4.4 gst_tensor_sparse_dec_class_init()

```
static void gst_tensor_sparse_dec_class_init (
            GstTensorSparseDecClass * klass )  [static]
```

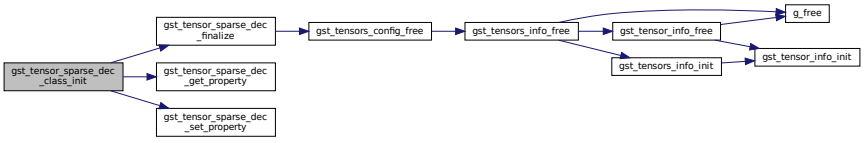
Initialize the tensor_sparse's class.

GstTensorSparseDec::silent:

The flag to enable/disable debugging messages.

Definition at line 101 of file gsttensor_sparsedec.c.

Here is the call graph for this function:



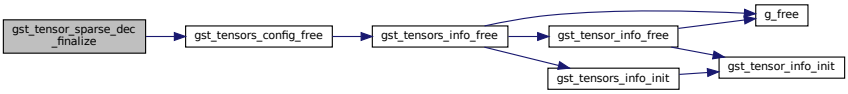
9.56.4.5 gst_tensor_sparse_dec_finalize()

```
static void gst_tensor_sparse_dec_finalize (
            GObject * object )  [static]
```

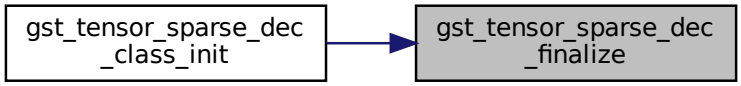
Function to finalize instance.

Definition at line 172 of file gsttensor_sparsedec.c.

Here is the call graph for this function:



Here is the caller graph for this function:



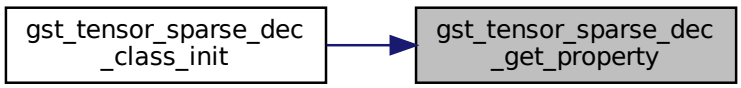
9.56.4.6 gst_tensor_sparse_dec_get_property()

```
static void gst_tensor_sparse_dec_get_property (
             GObject * object,
             guint prop_id,
             GValue * value,
             GParamSpec * pspec )  [static]
```

Getter for tensor_sparse_dec properties.

Definition at line 208 of file gsttensor_sparsedec.c.

Here is the caller graph for this function:



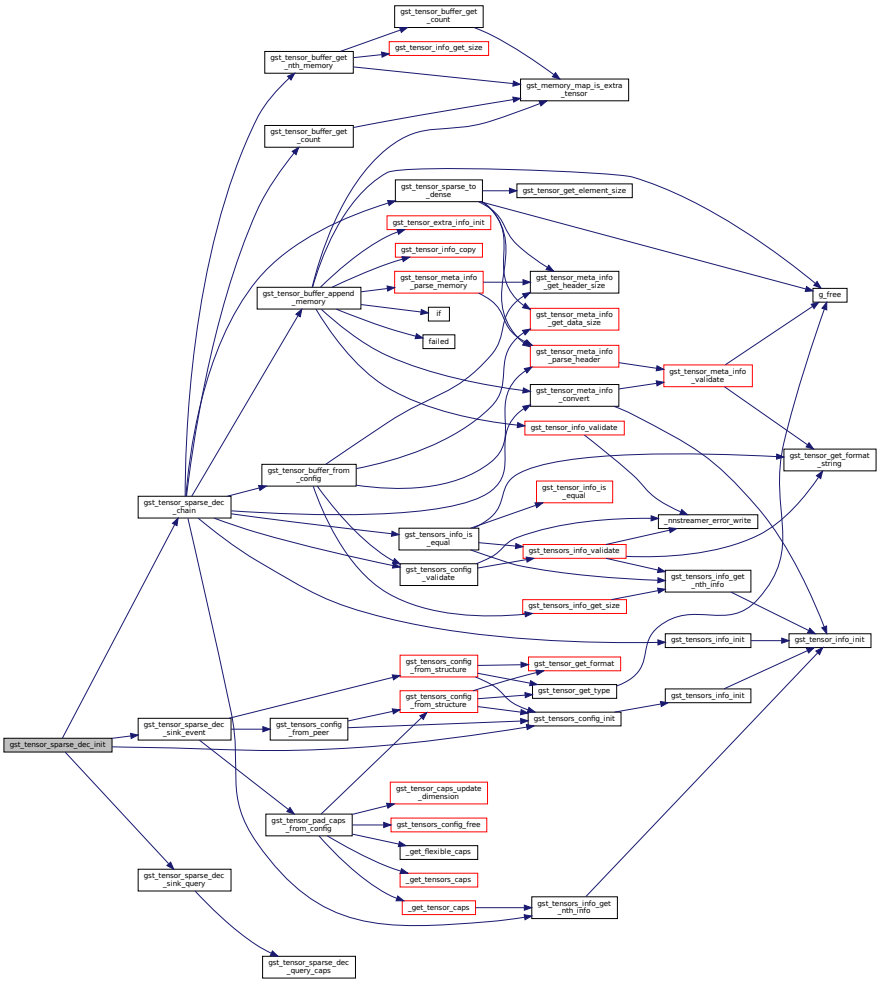
9.56.4.7 gst_tensor_sparse_dec_init()

```
static void gst_tensor_sparse_dec_init (
            GstTensorSparseDec * self )  [static]
```

Initialize tensor_sparse_dec element.

Definition at line 141 of file gsttensor_sparsedec.c.

Here is the call graph for this function:



9.56.4.8 gst_tensor_sparse_dec_query_caps()

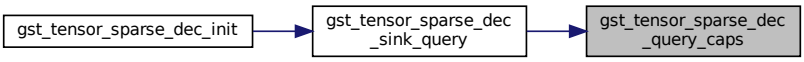
```
static GstCaps* gst_tensor_sparse_dec_query_caps (
            GstTensorSparseDec * self,
            GstPad * pad,
            GstCaps * filter )  [static]
```

Get pad caps for caps negotiation.

pad don't have current caps. use the template caps

Definition at line 229 of file gsttensor_sparsedec.c.

Here is the caller graph for this function:



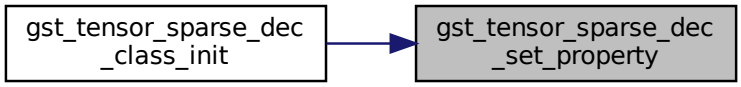
9.56.4.9 gst_tensor_sparse_dec_set_property()

```
static void gst_tensor_sparse_dec_set_property (
            GObject * object,
            guint prop_id,
            const GValue * value,
            GParamSpec * pspec )  [static]
```

Setter for tensor_sparse_dec properties.

Definition at line 187 of file gsttensor_sparsedec.c.

Here is the caller graph for this function:



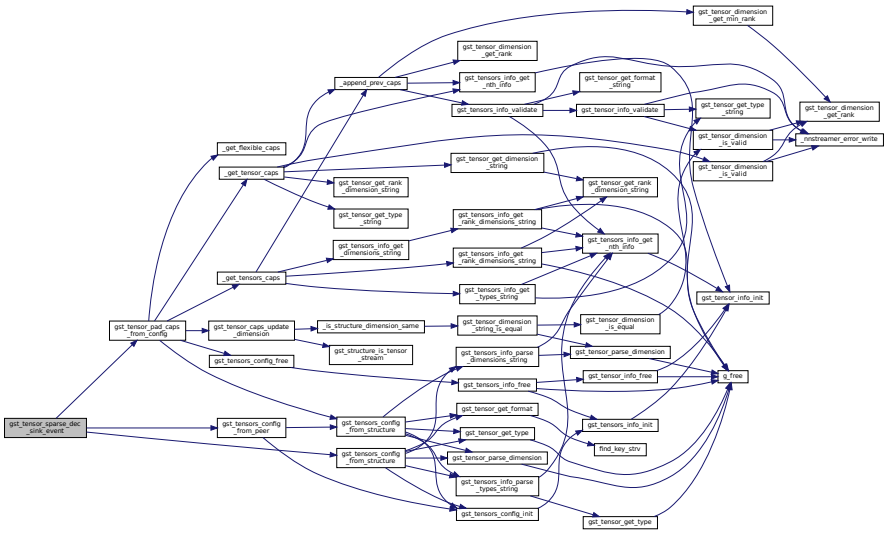
9.56.4.10 gst_tensor_sparse_dec_sink_event()

```
static gboolean gst_tensor_sparse_dec_sink_event (
            GstPad * pad,
            GstObject * parent,
            GstEvent * event )  [static]
```

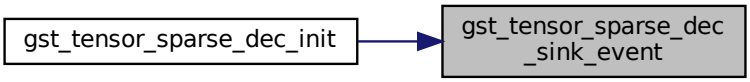
This function handles sink pad event.

Definition at line 313 of file gsttensor_sparsedec.c.

Here is the call graph for this function:



Here is the caller graph for this function:



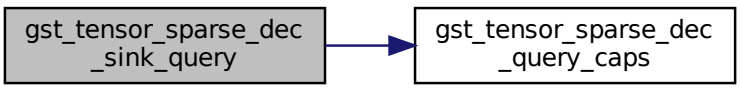
9.56.4.11 gst_tensor_sparse_dec_sink_query()

```
static gboolean gst_tensor_sparse_dec_sink_query (
            GstPad * pad,
            GstObject * parent,
            GstQuery * query )  [static]
```

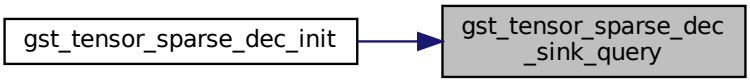
This function handles sink pad query.

Definition at line 260 of file gsttensor_sparsedec.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.56.5 Variable Documentation

9.56.5.1 sink_template

```
GstStaticPadTemplate sink_template  [static]
```

Initial value:

```
= GST_STATIC_PAD_TEMPLATE ("sink",
    GST_PAD_SINK,
    GST_PAD_ALWAYS,
    GST_STATIC_CAPS (GST_TENSORS_SPARSE_CAP_DEFAULT))
```

Template for sink pad.

Definition at line 68 of file gsttensor_sparsedec.c.

9.56.5.2 src_template

```
GstStaticPadTemplate src_template  [static]
```

Initial value:

```
= GST_STATIC_PAD_TEMPLATE ("src",
    GST_PAD_SRC,
    GST_PAD_ALWAYS,
    GST_STATIC_CAPS (GST_TENSORS_CAP_DEFAULT))
```

Template for src pad.

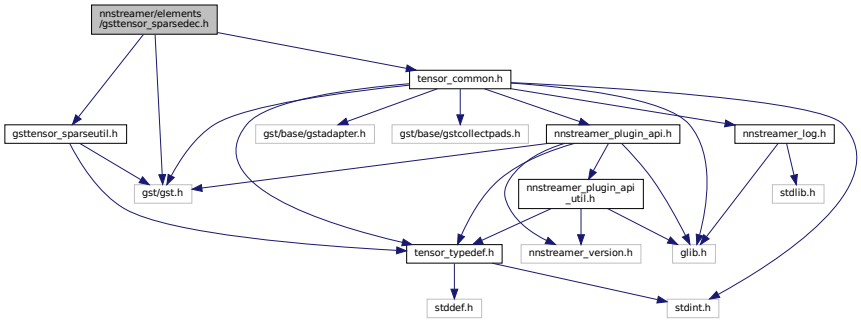
Definition at line 76 of file gsttensor_sparsedec.c.

9.57 nnstreamer/elements/gsttensor_sparsedec.h File Reference

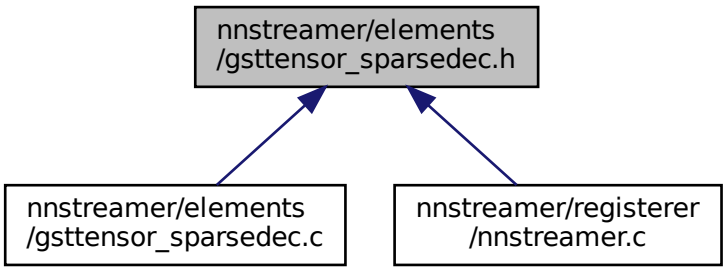
GStreamer element to decode sparse tensors into dense tensors.

```
#include <gst/gst.h>
#include <tensor_common.h>
#include "gsttensor_sparseutil.h"
```

Include dependency graph for gsttensor_sparsedec.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct _GstTensorSparseDec

  GstTensorSparseDec data structure.
- struct _GstTensorSparseDecClass

  GstTensorSparseClass data structure.

Macros

- #define GST_TYPE_TENSOR_SPARSE_DEC (gst_tensor_sparse_dec_get_type())
- #define GST_TENSOR_SPARSE_DEC(obj) (G_TYPE_CHECK_INSTANCE_CAST((obj),GST_TYPE_TENSOR_SPARSE_DE
- #define GST_TENSOR_SPARSE_DEC_CLASS(klass) (G_TYPE_CHECK_CLASS_CAST((klass),GST_TYPE_TENSOR_SPA
- #define GST_IS_TENSOR_SPARSE_DEC(obj) (G_TYPE_CHECK_INSTANCE_TYPE((obj),GST_TYPE_TENSOR_SPARSE_
- #define GST_IS_TENSOR_SPARSE_DEC_CLASS(klass) (G_TYPE_CHECK_CLASS_TYPE((klass),GST_TYPE_TENSOR_S

Typedefs

- typedef struct _GstTensorSparseDec GstTensorSparseDec
- typedef struct _GstTensorSparseDecClass GstTensorSparseDecClass

Functions

- GType gst_tensor_sparse_dec_get_type (void)

  Function to get type of tensor_sparse.

9.57.1 Detailed Description

GStreamer element to decode sparse tensors into dense tensors.

Date

27 Jul 2021

See also

https://github.com/nnstreamer/nnstreamer

Author

Yongjoo Ahn yongjoo1.ahn@samsung.com

Bug No known bugs except for NYI items

9.57.2 Macro Definition Documentation

9.57.2.1 GST_IS_TENSOR_SPARSE_DEC

```
#define GST_IS_TENSOR_SPARSE_DEC(
              obj ) (G_TYPE_CHECK_INSTANCE_TYPE((obj),GST_TYPE_TENSOR_SPARSE_DEC))
```

Definition at line 28 of file gsttensor_sparsedec.h.

9.57.2.2 GST_IS_TENSOR_SPARSE_DEC_CLASS

```
#define GST_IS_TENSOR_SPARSE_DEC_CLASS(
              klass ) (G_TYPE_CHECK_CLASS_TYPE((klass),GST_TYPE_TENSOR_SPARSE_DEC))
```

Definition at line 30 of file gsttensor_sparsedec.h.

9.57.2.3 GST_TENSOR_SPARSE_DEC

```
#define GST_TENSOR_SPARSE_DEC(
              obj ) (G_TYPE_CHECK_INSTANCE_CAST((obj),GST_TYPE_TENSOR_SPARSE_DEC,GstTensorSparseDec))
```

Definition at line 24 of file gsttensor_sparsedec.h.

9.57.2.4 GST_TENSOR_SPARSE_DEC_CLASS

```
#define GST_TENSOR_SPARSE_DEC_CLASS(
              klass ) (G_TYPE_CHECK_CLASS_CAST((klass),GST_TYPE_TENSOR_SPARSE_DEC,GstTensorSparseDecClass))
```

Definition at line 26 of file gsttensor_sparsedec.h.

9.57.2.5 GST_TYPE_TENSOR_SPARSE_DEC

```
#define GST_TYPE_TENSOR_SPARSE_DEC (gst_tensor_sparse_dec_get_type())
```

Definition at line 22 of file gsttensor_sparsedec.h.

9.57.3 Typedef Documentation

9.57.3.1 GstTensorSparseDec

```
typedef struct _GstTensorSparseDec GstTensorSparseDec
```

Definition at line 33 of file gsttensor_sparsedec.h.

9.57.3.2 GstTensorSparseDecClass

```
typedef struct _GstTensorSparseDecClass GstTensorSparseDecClass
```

Definition at line 34 of file gsttensor_sparsedec.h.

9.57.4 Function Documentation

9.57.4.1 gst_tensor_sparse_dec_get_type()

```
GType gst_tensor_sparse_dec_get_type (
            void  )
```

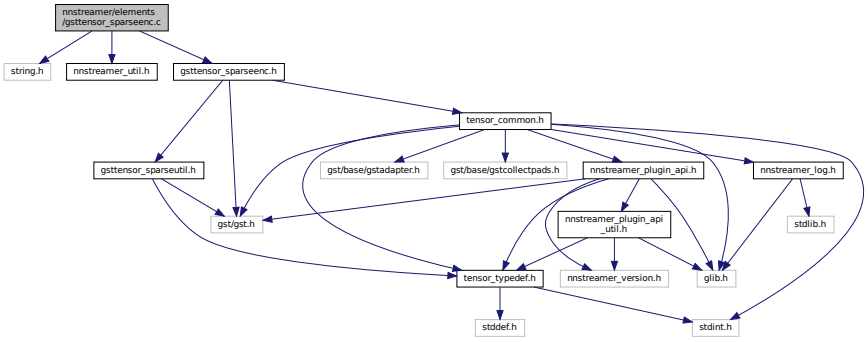
Function to get type of tensor_sparse.

9.58 nnstreamer/elements/gsttensor_sparseenc.c File Reference

GStreamer element to encode dense tensors into sparse tensors.

```
#include <string.h>
#include <nnstreamer_util.h>
#include "gsttensor_sparseenc.h"
```

Include dependency graph for gsttensor_sparseenc.c:



Macros

- #define DBG (!self->silent)

  Macro for debug mode.
- #define GST_CAT_DEFAULT gst_tensor_sparse_enc_debug
- #define DEFAULT_SILENT TRUE

  Flag to print minimized log.
- #define gst_tensor_sparse_enc_parent_class parent_class

Enumerations

- enum { PROP_0, PROP_SILENT }

  tensor_sparse_enc properties

Functions

- GST_DEBUG_CATEGORY_STATIC (gst_tensor_sparse_enc_debug)
- G_DEFINE_TYPE (GstTensorSparseEnc, gst_tensor_sparse_enc, GST_TYPE_ELEMENT)
- static void gst_tensor_sparse_enc_finalize (GObject *object)

  Function to finalize instance.
- static void gst_tensor_sparse_enc_set_property (GObject *object, guint prop_id, const GValue *value, GParamSpec *pspec)

  Setter for tensor_sparse_enc properties.
- static void gst_tensor_sparse_enc_get_property (GObject *object, guint prop_id, GValue *value, GParamSpec *pspec)

  Getter for tensor_sparse_enc properties.
- static GstFlowReturn gst_tensor_sparse_enc_chain (GstPad *pad, GstObject *parent, GstBuffer *buf)

  Internal function to transform the input buffer.
- static GstCaps * gst_tensor_sparse_enc_query_caps (GstTensorSparseEnc *self, GstPad *pad, GstCaps *filter)

  Get pad caps for caps negotiation.
- static gboolean gst_tensor_sparse_enc_sink_event (GstPad *pad, GstObject *parent, GstEvent *event)

  This function handles sink pad event.
- static gboolean gst_tensor_sparse_enc_sink_query (GstPad *pad, GstObject *parent, GstQuery *query)

  This function handles sink pad query.
- static void gst_tensor_sparse_enc_class_init (GstTensorSparseEncClass *klass)

  Initialize the tensor_sparse's class.
- static void gst_tensor_sparse_enc_init (GstTensorSparseEnc *self)

  Initialize tensor_sparse_enc element.
- static gboolean gst_tensor_sparse_enc_parse_caps (GstTensorSparseEnc *self, const GstCaps *caps)

  Parse caps and set tensors config.

Variables

- static GstStaticPadTemplate sink_template

  Template for sink pad.
- static GstStaticPadTemplate src_template

  Template for src pad.

9.58.1 Detailed Description

GStreamer element to encode dense tensors into sparse tensors.

Date

27 Jul 2021

See also

https://github.com/nnstreamer/nnstreamer

Author

Yongjoo Ahn yongjoo1.ahn@samsung.com

Bug No known bugs except for NYI items

9.58.2 Macro Definition Documentation

9.58.2.1 DBG

```
#define DBG (!self->silent)
```

Macro for debug mode.

SECTION:element-tensor_sparse_enc

tensor_sparse_enc is a GStreamer element to encode incoming tensor into sparse format.

The input is always in the format of other/tensors,format=static. The output is always in the format of ohter/tensors,format=sparse.

Please see also tensor_sparse_dec.

<refsect2> <title>Example launch line</title> |[gst-launch-1.0 ... ! other/tensors,format=static ! \ tensor_↩ sparse_enc ! tensor_sink]| </refsect2>

Definition at line 44 of file gsttensor_sparseenc.c.

9.58.2.2 DEFAULT_SILENT

```
#define DEFAULT_SILENT TRUE
```

Flag to print minimized log.

Definition at line 62 of file gsttensor_sparseenc.c.

9.58.2.3 GST_CAT_DEFAULT

```
#define GST_CAT_DEFAULT gst_tensor_sparse_enc_debug
```

Definition at line 48 of file gsttensor_sparseenc.c.

9.58.2.4 gst_tensor_sparse_enc_parent_class

```
#define gst_tensor_sparse_enc_parent_class parent_class
```

Definition at line 80 of file gsttensor_sparseenc.c.

9.58.3 Enumeration Type Documentation

9.58.3.1 anonymous enum

```
anonymous enum
```

tensor_sparse_enc properties

Enumerator

PROP_0	
PROP_SILENT	

Definition at line 53 of file gsttensor_sparseenc.c.

9.58.4 Function Documentation

9.58.4.1 G_DEFINE_TYPE()

```
G_DEFINE_TYPE (
            GstTensorSparseEnc ,
            gst_tensor_sparse_enc ,
            GST_TYPE_ELEMENT  )
```

9.58.4.2 GST_DEBUG_CATEGORY_STATIC()

```
GST_DEBUG_CATEGORY_STATIC (
            gst_tensor_sparse_enc_debug  )
```

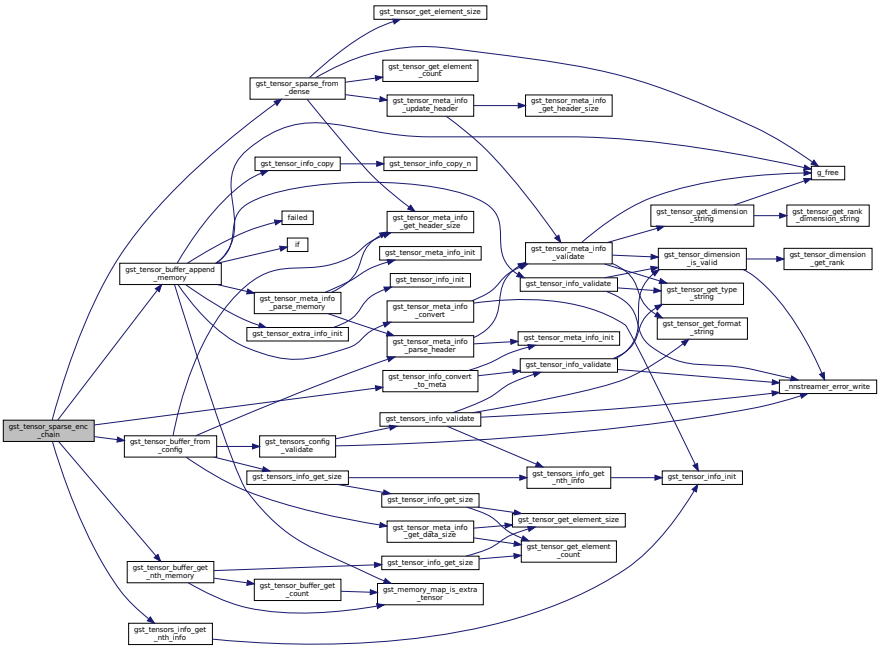
9.58.4.3 gst_tensor_sparse_enc_chain()

```
static GstFlowReturn gst_tensor_sparse_enc_chain (
            GstPad * pad,
            GstObject * parent,
            GstBuffer * buf )  [static]
```

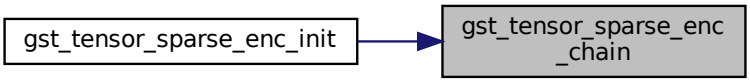
Internal function to transform the input buffer.

Definition at line 372 of file gsttensor_sparseenc.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.58.4.4 gst_tensor_sparse_enc_class_init()

```
static void gst_tensor_sparse_enc_class_init (
            GstTensorSparseEncClass * klass )  [static]
```

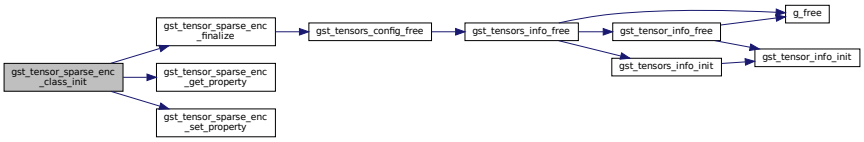
Initialize the tensor_sparse's class.

GstTensorSparseEnc::silent:

The flag to enable/disable debugging messages.

Definition at line 101 of file gsttensor_sparseenc.c.

Here is the call graph for this function:



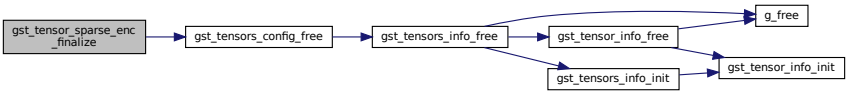
9.58.4.5 gst_tensor_sparse_enc_finalize()

```
static void gst_tensor_sparse_enc_finalize (
            GObject * object )  [static]
```

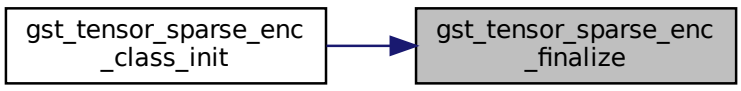
Function to finalize instance.

Definition at line 172 of file gsttensor_sparseenc.c.

Here is the call graph for this function:



Here is the caller graph for this function:



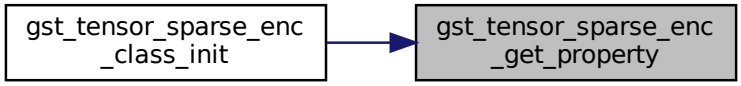
9.58.4.6 gst_tensor_sparse_enc_get_property()

```
static void gst_tensor_sparse_enc_get_property (
            GObject * object,
            guint prop_id,
            GValue * value,
            GParamSpec * pspec )  [static]
```

Getter for tensor_sparse_enc properties.

Definition at line 207 of file gsttensor_sparseenc.c.

Here is the caller graph for this function:



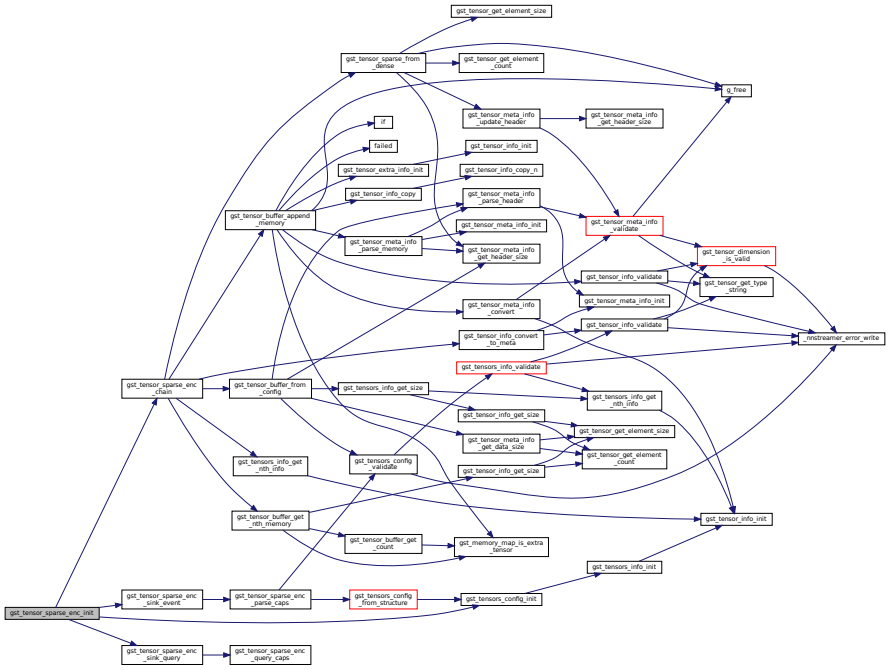
9.58.4.7 gst_tensor_sparse_enc_init()

```
static void gst_tensor_sparse_enc_init (
            GstTensorSparseEnc * self )  [static]
```

Initialize tensor_sparse_enc element.

Definition at line 142 of file gsttensor_sparseenc.c.

Here is the call graph for this function:



9.58.4.8 gst_tensor_sparse_enc_parse_caps()

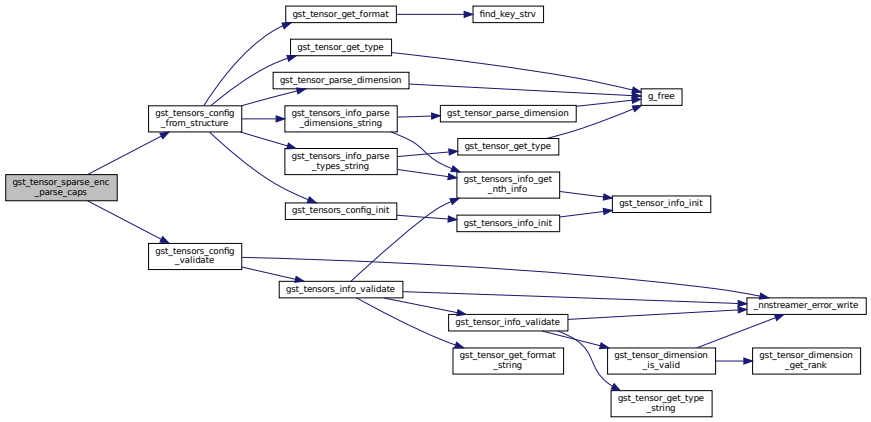
```
static gboolean gst_tensor_sparse_enc_parse_caps (
            GstTensorSparseEnc * self,
            const GstCaps * caps )  [static]
```

Parse caps and set tensors config.

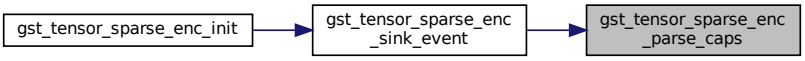
not fully configured

Definition at line 228 of file gsttensor_sparseenc.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.58.4.9 gst_tensor_sparse_enc_query_caps()

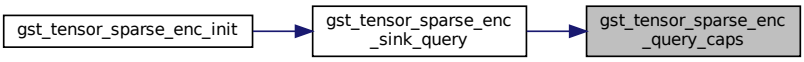
```
static GstCaps * gst_tensor_sparse_enc_query_caps (
            GstTensorSparseEnc * self,
            GstPad * pad,
            GstCaps * filter )  [static]
```

Get pad caps for caps negotiation.

pad don't have current caps. use the template caps

Definition at line 286 of file gsttensor_sparseenc.c.

Here is the caller graph for this function:



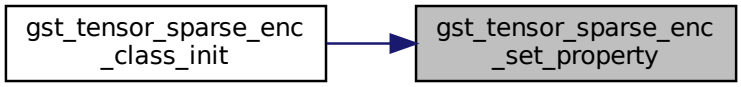
9.58.4.10 gst_tensor_sparse_enc_set_property()

```
static void gst_tensor_sparse_enc_set_property (
            GObject * object,
            guint prop_id,
            const GValue * value,
            GParamSpec * pspec )  [static]
```

Setter for tensor_sparse_enc properties.

Definition at line 186 of file gsttensor_sparseenc.c.

Here is the caller graph for this function:



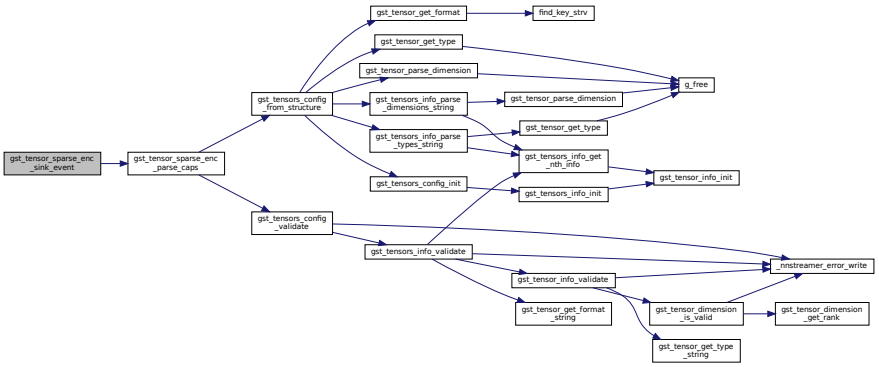
9.58.4.11 gst_tensor_sparse_enc_sink_event()

```
static gboolean gst_tensor_sparse_enc_sink_event (
            GstPad * pad,
            GstObject * parent,
            GstEvent * event )  [static]
```

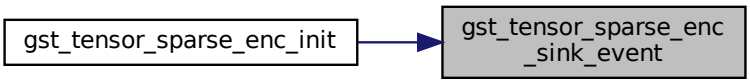
This function handles sink pad event.

Definition at line 254 of file gsttensor_sparseenc.c.

Here is the call graph for this function:



Here is the caller graph for this function:



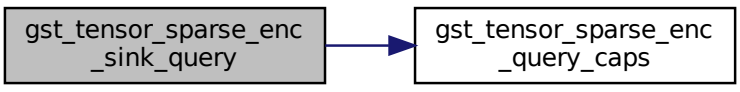
9.58.4.12 gst_tensor_sparse_enc_sink_query()

```
static gboolean gst_tensor_sparse_enc_sink_query (
            GstPad * pad,
            GstObject * parent,
            GstQuery * query )  [static]
```

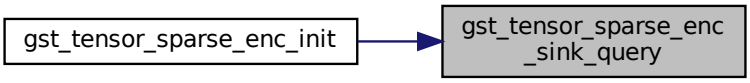
This function handles sink pad query.

Definition at line 317 of file gsttensor_sparseenc.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.58.5 Variable Documentation

9.58.5.1 sink_template

```
GstStaticPadTemplate sink_template  [static]
```

Initial value:

```
= GST_STATIC_PAD_TEMPLATE ("sink",
    GST_PAD_SINK,
    GST_PAD_ALWAYS,
    GST_STATIC_CAPS (GST_TENSORS_CAP_DEFAULT))
```

Template for sink pad.

Definition at line 67 of file gsttensor_sparseenc.c.

9.58.5.2 src_template

```
GstStaticPadTemplate src_template  [static]
```

Initial value:

```
= GST_STATIC_PAD_TEMPLATE ("src",
    GST_PAD_SRC,
    GST_PAD_ALWAYS,
    GST_STATIC_CAPS (GST_TENSORS_SPARSE_CAP_DEFAULT))
```

Template for src pad.

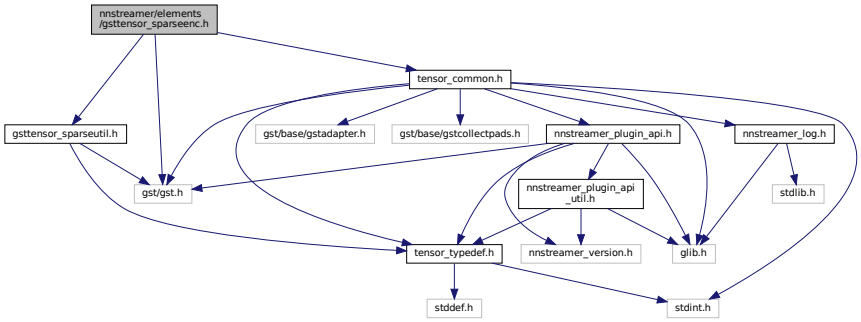
Definition at line 75 of file gsttensor_sparseenc.c.

9.59 nnstreamer/elements/gsttensor_sparseenc.h File Reference

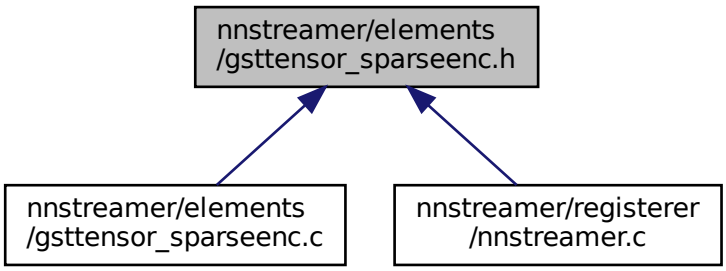
GStreamer element to encode sparse tensors into dense tensors.

```
#include <gst/gst.h>
#include <tensor_common.h>
#include "gsttensor_sparseutil.h"
```

Include dependency graph for gsttensor_sparseenc.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct _GstTensorSparseEnc

  GstTensorSparseEnc data structure.
- struct _GstTensorSparseEncClass

  GstTensorSparseClass data structure.

Macros

- #define GST_TYPE_TENSOR_SPARSE_ENC (gst_tensor_sparse_enc_get_type())
- #define GST_TENSOR_SPARSE_ENC(obj) (G_TYPE_CHECK_INSTANCE_CAST((obj),GST_TYPE_TENSOR_SPARSE_EN
- #define GST_TENSOR_SPARSE_ENC_CLASS(klass) (G_TYPE_CHECK_CLASS_CAST((klass),GST_TYPE_TENSOR_SPA
- #define GST_IS_TENSOR_SPARSE_ENC(obj) (G_TYPE_CHECK_INSTANCE_TYPE((obj),GST_TYPE_TENSOR_SPARSE_
- #define GST_IS_TENSOR_SPARSE_ENC_CLASS(klass) (G_TYPE_CHECK_CLASS_TYPE((klass),GST_TYPE_TENSOR_S

Typedefs

- typedef struct _GstTensorSparseEnc GstTensorSparseEnc
- typedef struct _GstTensorSparseEncClass GstTensorSparseEncClass

Functions

- GType gst_tensor_sparse_enc_get_type (void)

  Function to get type of tensor_sparse.

9.59.1 Detailed Description

GStreamer element to encode sparse tensors into dense tensors.

Date

27 Jul 2021

See also

https://github.com/nnstreamer/nnstreamer

Author

Yongjoo Ahn yongjoo1.ahn@samsung.com

Bug No known bugs except for NYI items

9.59.2 Macro Definition Documentation

9.59.2.1 GST_IS_TENSOR_SPARSE_ENC

```
#define GST_IS_TENSOR_SPARSE_ENC(
              obj ) (G_TYPE_CHECK_INSTANCE_TYPE((obj),GST_TYPE_TENSOR_SPARSE_ENC))
```

Definition at line 28 of file gsttensor_sparseenc.h.

9.59.2.2 GST_IS_TENSOR_SPARSE_ENC_CLASS

```
#define GST_IS_TENSOR_SPARSE_ENC_CLASS(
              klass ) (G_TYPE_CHECK_CLASS_TYPE((klass),GST_TYPE_TENSOR_SPARSE_ENC))
```

Definition at line 30 of file gsttensor_sparseenc.h.

9.59.2.3 GST_TENSOR_SPARSE_ENC

```
#define GST_TENSOR_SPARSE_ENC(
              obj ) (G_TYPE_CHECK_INSTANCE_CAST((obj),GST_TYPE_TENSOR_SPARSE_ENC,GstTensorSparseEnc))
```

Definition at line 24 of file gsttensor_sparseenc.h.

9.59.2.4 GST_TENSOR_SPARSE_ENC_CLASS

```
#define GST_TENSOR_SPARSE_ENC_CLASS(
              klass ) (G_TYPE_CHECK_CLASS_CAST((klass),GST_TYPE_TENSOR_SPARSE_ENC,GstTensorSparseEncClass))
```

Definition at line 26 of file gsttensor_sparseenc.h.

9.59.2.5 GST_TYPE_TENSOR_SPARSE_ENC

```
#define GST_TYPE_TENSOR_SPARSE_ENC (gst_tensor_sparse_enc_get_type())
```

Definition at line 22 of file gsttensor_sparseenc.h.

9.59.3 Typedef Documentation

9.59.3.1 GstTensorSparseEnc

```
typedef struct _GstTensorSparseEnc GstTensorSparseEnc
```

Definition at line 33 of file gsttensor_sparseenc.h.

9.59.3.2 GstTensorSparseEncClass

```
typedef struct _GstTensorSparseEncClass GstTensorSparseEncClass
```

Definition at line 34 of file gsttensor_sparseenc.h.

9.59.4 Function Documentation

9.59.4.1 gst_tensor_sparse_enc_get_type()

```
GType gst_tensor_sparse_enc_get_type (
            void  )
```

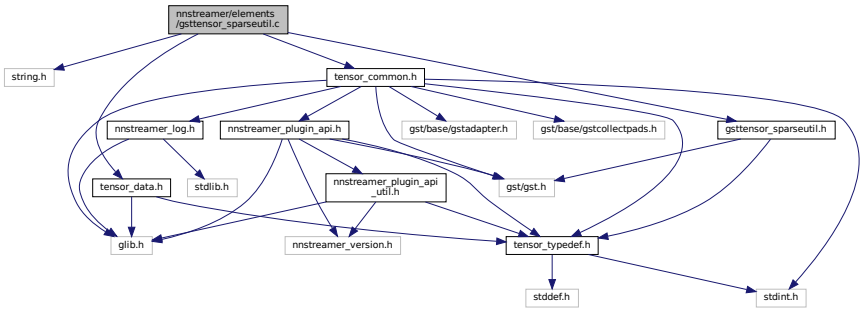
Function to get type of tensor_sparse.

9.60 nnstreamer/elements/gsttensor_sparseutil.c File Reference

Util functions for tensor_sparse encoder and decoder.

```
#include <string.h>
#include <tensor_common.h>
#include <tensor_data.h>
#include "gsttensor_sparseutil.h"
```

Include dependency graph for gsttensor_sparseutil.c:



Functions

- GstMemory * gst_tensor_sparse_to_dense (GstTensorMetaInfo *meta, GstMemory *mem)

  Make dense tensor with input sparse tensor.
- GstMemory * gst_tensor_sparse_from_dense (GstTensorMetaInfo *meta, GstMemory *mem)

  Make sparse tensor with input dense tensor.

9.60.1 Detailed Description

Util functions for tensor_sparse encoder and decoder.

GStreamer / NNStreamer Sparse Tensor support Copyright (C) 2021 Yongjoo Ahn yongjoo1.↩ahn@samsung.com

Date

27 Jul 2021

See also

https://github.com/nnstreamer/nnstreamer

Author

Yongjoo Ahn yongjoo1.ahn@samsung.com

Bug No known bugs except for NYI items

9.60.2 Function Documentation

9.60.2.1 gst_tensor_sparse_from_dense()

```
GstMemory* gst_tensor_sparse_from_dense (
            GstTensorMetaInfo * meta,
            GstMemory * mem )
```

Make sparse tensor with input dense tensor.

Parameters

in,out	*meta*	tensor meta structure to be updated
in	*mem*	gst-memory of dense tensor data

Returns

pointer of GstMemory with sparse tensor data or NULL on error. Caller should handle this newly allocated memory.

alloc maximum possible size of memory

Consider using macro to reduce loc and readability

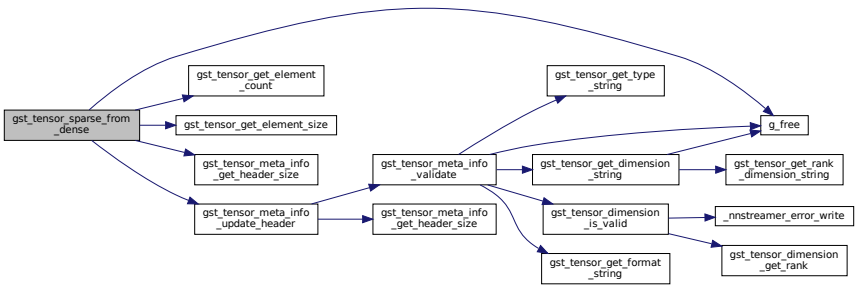
update meta nnz info

write to output buffer

add meta info header

Definition at line 116 of file gsttensor_sparseutil.c.

Here is the call graph for this function:



Here is the caller graph for this function:

9.60.2.2 gst_tensor_sparse_to_dense()

```
GstMemory* gst_tensor_sparse_to_dense (
            GstTensorMetaInfo * meta,
            GstMemory * mem )
```

Make dense tensor with input sparse tensor.

Parameters

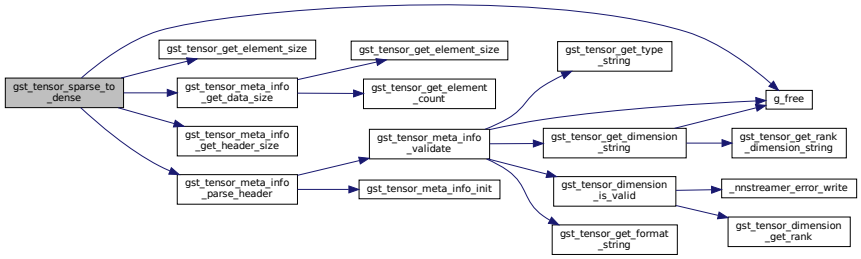
in,out	*meta*	tensor meta structure to be updated
in	*mem*	gst-memory of sparse tensor data

Returns

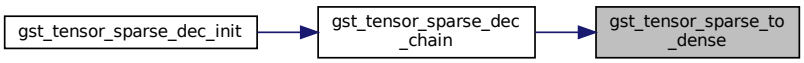
pointer of GstMemory with dense tensor data or NULL on error. Caller should handle this newly allocated memory.

Definition at line 27 of file gsttensor_sparseutil.c.

Here is the call graph for this function:



Here is the caller graph for this function:

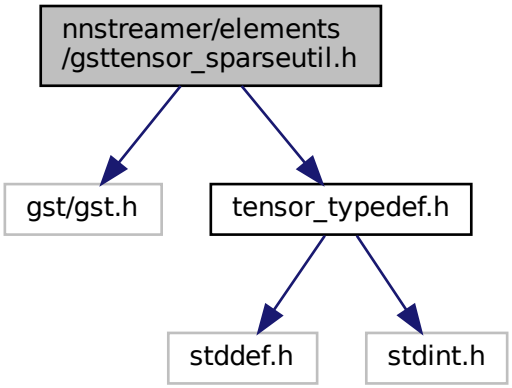


9.61 nnstreamer/elements/gsttensor_sparseutil.h File Reference

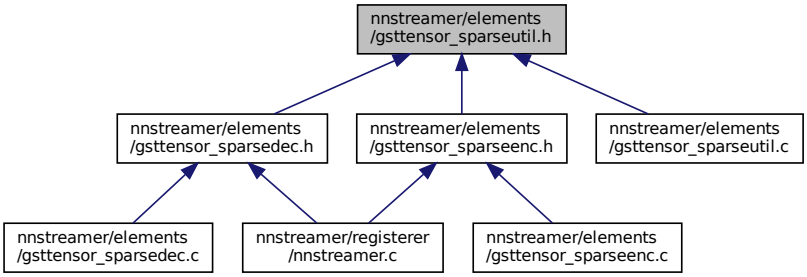
Util functions for tensor_sparse encoder and decoder.

```
#include <gst/gst.h>
#include <tensor_typedef.h>
```

Include dependency graph for gsttensor_sparseutil.h:



This graph shows which files directly or indirectly include this file:



Functions

- G_BEGIN_DECLS GstMemory * gst_tensor_sparse_to_dense (GstTensorMetaInfo *meta, GstMemory *mem)

  Make dense tensor with input sparse tensor.

- GstMemory * gst_tensor_sparse_from_dense (GstTensorMetaInfo *meta, GstMemory *mem)

  Make sparse tensor with input dense tensor.

9.61.1 Detailed Description

Util functions for tensor_sparse encoder and decoder.

GStreamer / NNStreamer Sparse Tensor support Copyright (C) 2021 Yongjoo Ahn yongjoo1.↩ ahn@samsung.com

Date

06 Jul 2021

See also

https://github.com/nnstreamer/nnstreamer

Author

Yongjoo Ahn yongjoo1.ahn@samsung.com

Bug No known bugs except for NYI items

9.61.2 Function Documentation

9.61.2.1 gst_tensor_sparse_from_dense()

```
GstMemory* gst_tensor_sparse_from_dense (
            GstTensorMetaInfo * meta,
            GstMemory * mem )
```

Make sparse tensor with input dense tensor.

Parameters

in,out	*meta*	tensor meta structure to be updated
in	*mem*	gst-memory of dense tensor data

Returns

pointer of GstMemory with sparse tensor data or NULL on error. Caller should handle this newly allocated memory.

alloc maximum possible size of memory

Consider using macro to reduce loc and readability

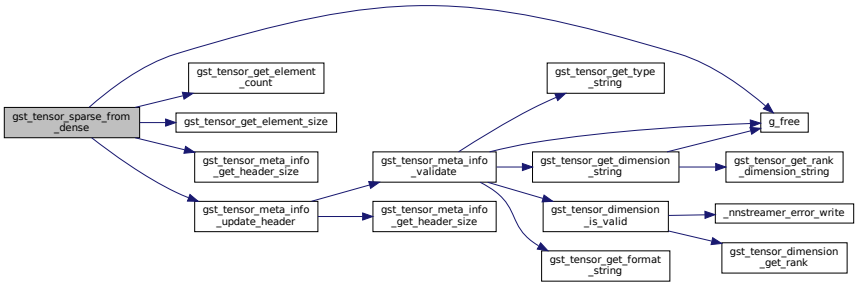
update meta nnz info

write to output buffer

add meta info header

Definition at line 116 of file gsttensor_sparseutil.c.

Here is the call graph for this function:



Here is the caller graph for this function:

9.61.2.2 gst_tensor_sparse_to_dense()

```
G_BEGIN_DECLS GstMemory* gst_tensor_sparse_to_dense (
            GstTensorMetaInfo * meta,
            GstMemory * mem )
```

Make dense tensor with input sparse tensor.

Parameters

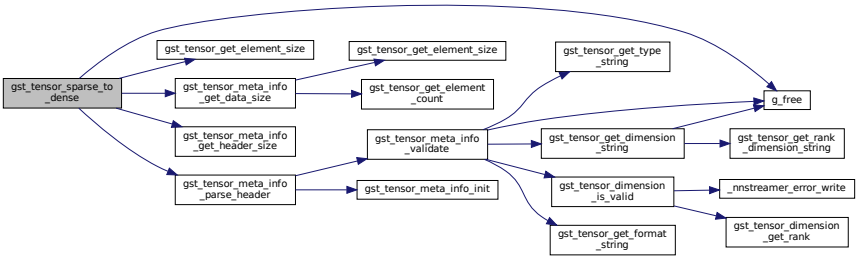
in,out	*meta*	tensor meta structure to be updated
in	*mem*	gst-memory of sparse tensor data

Returns

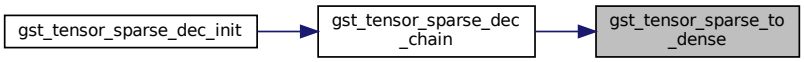
pointer of GstMemory with dense tensor data or NULL on error. Caller should handle this newly allocated memory.

Definition at line 27 of file gsttensor_sparseutil.c.

Here is the call graph for this function:



Here is the caller graph for this function:

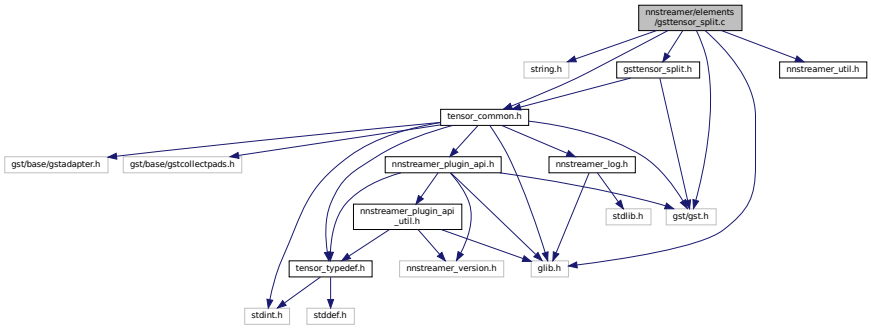


9.62 nnstreamer/elements/gsttensor_split.c File Reference

GStreamer plugin to split tensor (as a filter for other general neural network filters)

```
#include <string.h>
#include <gst/gst.h>
#include <glib.h>
#include "gsttensor_split.h"
#include <tensor_common.h>
#include <nnstreamer_util.h>
```

Include dependency graph for gsttensor_split.c:



Macros

- #define GST_CAT_DEFAULT gst_tensor_split_debug
- #define CAPS_STRING GST_TENSOR_CAP_DEFAULT ";" GST_TENSORS_CAP_WITH_NUM ("1")

  Template caps string.
- #define gst_tensor_split_parent_class parent_class

Enumerations

- enum { PROP_0, PROP_SILENT, PROP_TENSORPICK, PROP_TENSORSEG }

Functions

- GST_DEBUG_CATEGORY_STATIC (gst_tensor_split_debug)
- static GstFlowReturn gst_tensor_split_chain (GstPad *pad, GstObject *parent, GstBuffer *buf)

  chain function for sink (gst element vmethod)
- static gboolean gst_tensor_split_event (GstPad *pad, GstObject *parent, GstEvent *event)

  event function for sink (gst element vmethod)
- static GstStateChangeReturn gst_tensor_split_change_state (GstElement *element, GstStateChange transition)

  change state (gst element vmethod)
- static void gst_tensor_split_set_property (GObject *object, guint prop_id, const GValue *value, GParamSpec *pspec)

  Get property (gst element vmethod)
- static void gst_tensor_split_get_property (GObject *object, guint prop_id, GValue *value, GParamSpec *pspec)

  Get property (gst element vmethod)
- static void gst_tensor_split_finalize (GObject *object)

  finalize function for tensor split (gst element vmethod)
- G_DEFINE_TYPE (GstTensorSplit, gst_tensor_split, GST_TYPE_ELEMENT)
- static void gst_tensor_split_class_init (GstTensorSplitClass *klass)

  initialize the tensor_split's class
- static void gst_tensor_split_init (GstTensorSplit *split)

  initialize the new element instantiate pads and add them to element set pad callback functions initialize instance structure
- static void gst_tensor_split_remove_src_pads (GstTensorSplit *split)

  function to remove srcpad list
- static gboolean gst_tensor_split_get_capsparam (GstTensorSplit *split, GstCaps *caps)

  Set Caps in pad.
- static GstTensorPad * gst_tensor_split_get_tensor_pad (GstTensorSplit *split, GstBuffer *inbuf, gboolean *created, guint nth)

  Checking if the source pad is created and if not, create TensorPad.
- static GstFlowReturn gst_tensor_split_combine_flows (GstTensorSplit *split, GstTensorPad *pad, GstFlow↩ Return ret)

  Check the status among sources in split.
- static GstMemory * gst_tensor_split_get_splitted (GstTensorSplit *split, GstBuffer *buffer, gint nth)

  Make splitted tensor.
- static void _clear_tensorseg (tensor_dim **element)

  Glib Array Clear Function.

Variables

- static GstStaticPadTemplate src_templ

  the capabilities of the inputs and outputs. describe the real formats here.
- static GstStaticPadTemplate sink_templ

9.62.1 Detailed Description

GStreamer plugin to split tensor (as a filter for other general neural network filters)

GStreamer Copyright (C) 2005 Thomas Vander Stichele thomas@apestaart.org Copyright (C) 2005 Ronald S. Bultje rbultje@ronald.bitfreak.net Copyright (C) 2018 Jijoong Moon jijoong.moon@samsung.com

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Library General Public License as published by the Free Software Foundation; version 2.1 of the License.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Library General Public License for more details.

Date

27 Aug 2018

See also

https://github.com/nnstreamer/nnstreamer

Author

Jijoong Moon jijoong.moon@samsung.com

Bug No known bugs except for NYI items

9.62.2 Macro Definition Documentation

9.62.2.1 CAPS_STRING

```
#define CAPS_STRING GST_TENSOR_CAP_DEFAULT ";" GST_TENSORS_CAP_WITH_NUM ("1")
```

Template caps string.

Definition at line 73 of file gsttensor_split.c.

9.62.2.2 GST_CAT_DEFAULT

```
#define GST_CAT_DEFAULT gst_tensor_split_debug
```

Definition at line 60 of file gsttensor_split.c.

9.62.2.3 gst_tensor_split_parent_class

```
#define gst_tensor_split_parent_class parent_class
```

Definition at line 101 of file gsttensor_split.c.

9.62.3 Enumeration Type Documentation

9.62.3.1 anonymous enum

```
anonymous enum
```

Enumerator

PROP_0	
PROP_SILENT	
PROP_TENSORPICK	
PROP_TENSORSEG	

Definition at line 62 of file gsttensor_split.c.

9.62.4 Function Documentation

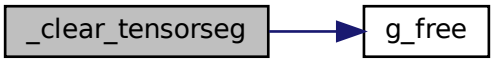
9.62.4.1 _clear_tensorseg()

```
static void _clear_tensorseg (
            tensor_dim ** element )  [static]
```

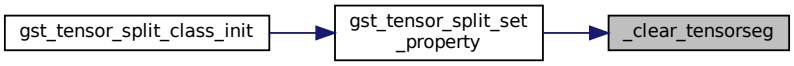
Glib Array Clear Function.

Definition at line 569 of file gsttensor_split.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.62.4.2 G_DEFINE_TYPE()

```
G_DEFINE_TYPE (
             GstTensorSplit ,
             gst_tensor_split ,
             GST_TYPE_ELEMENT  )
```

9.62.4.3 GST_DEBUG_CATEGORY_STATIC()

```
GST_DEBUG_CATEGORY_STATIC (
             gst_tensor_split_debug  )
```

SECTION:element-tensor_split

A Deuxer that split tensors stream to tensor stream for NN frameworks. The outputs are always in the format of other/tensor.

<refsect2> <title>Example launch line</title> |[gst-launch -v -m filesrc location=testcase_RGB_100x100.png ! pngdec ! videoscale ! imagefreeze ! videoconvert ! video/x-raw,format=RGB,width=100,height=100,framerate=0/1 ! tensor_converter ! tensor_split name=split tensorseg=2:100:100,1:100:100 split.src_0 ! queue ! filesink location=src0.log split.src_1 ! queue ! filesink location=src1.log]|

</refsect2>

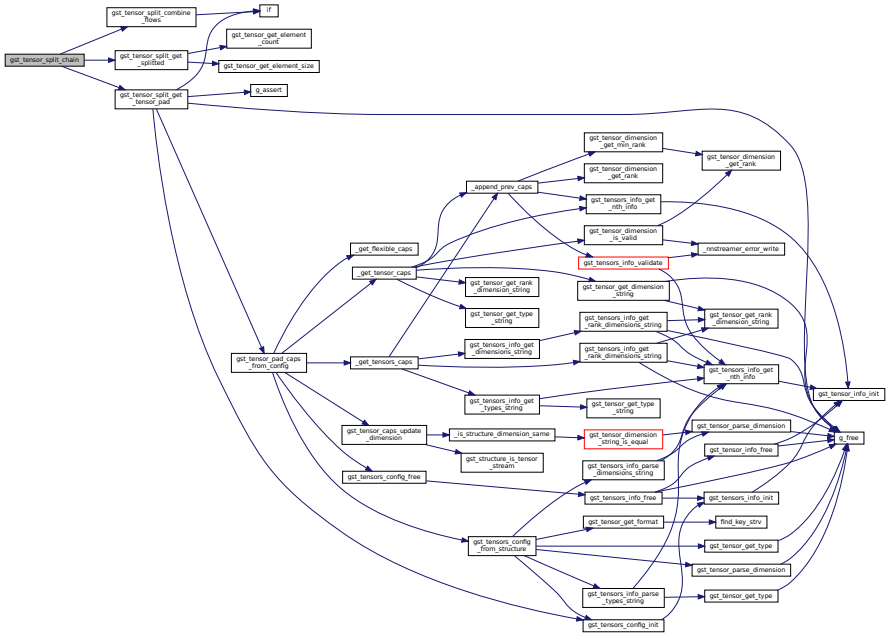
9.62.4.4 gst_tensor_split_chain()

```
static GstFlowReturn gst_tensor_split_chain (
             GstPad * pad,
             GstObject * parent,
             GstBuffer * buf )  [static]
```

chain function for sink (gst element vmethod)

Definition at line 459 of file gsttensor_split.c.

Here is the call graph for this function:



Here is the caller graph for this function:

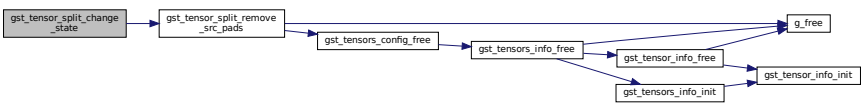
9.62.4.5 gst_tensor_split_change_state()

```
static GstStateChangeReturn gst_tensor_split_change_state (
            GstElement * element,
            GstStateChange transition )  [static]
```

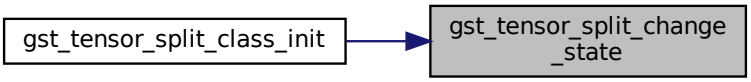
change state (gst element vmethod)

Definition at line 537 of file gsttensor_split.c.

Here is the call graph for this function:



Here is the caller graph for this function:



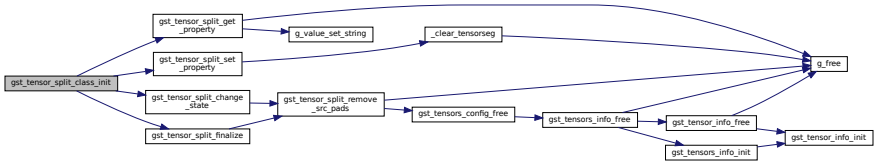
9.62.4.6 gst_tensor_split_class_init()

```
static void gst_tensor_split_class_init (
            GstTensorSplitClass * klass )  [static]
```

initialize the tensor_split's class

Definition at line 109 of file gsttensor_split.c.

Here is the call graph for this function:



9.62.4.7 gst_tensor_split_combine_flows()

```
static GstFlowReturn gst_tensor_split_combine_flows (
            GstTensorSplit * split,
            GstTensorPad * pad,
            GstFlowReturn ret )  [static]
```

Check the status among sources in split.

Parameters

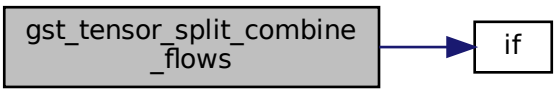
split	TensorSplit Object
TensorPad	Tensorpad
ret	return status of current pad

Returns

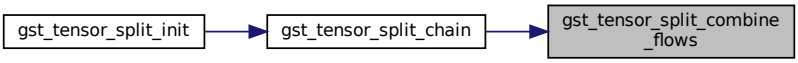
return status after check sources

Definition at line 387 of file gsttensor_split.c.

Here is the call graph for this function:



Here is the caller graph for this function:



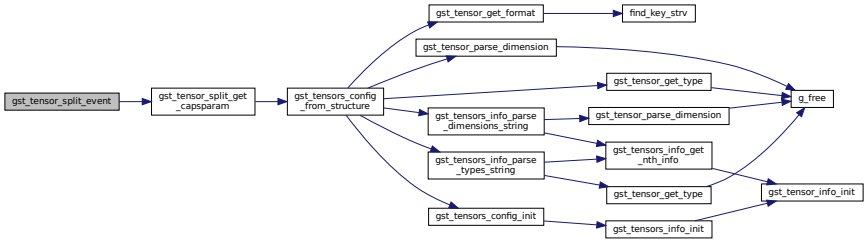
9.62.4.8 gst_tensor_split_event()

```
static gboolean gst_tensor_split_event (
            GstPad * pad,
            GstObject * parent,
            GstEvent * event )  [static]
```

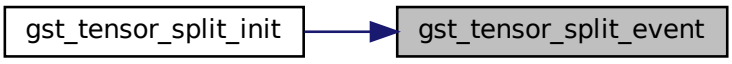
event function for sink (gst element vmethod)

Definition at line 233 of file gsttensor_split.c.

Here is the call graph for this function:



Here is the caller graph for this function:



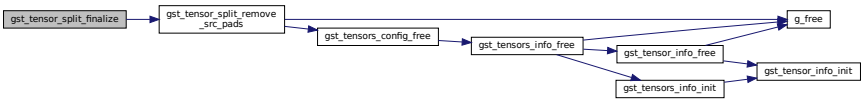
9.62.4.9 gst_tensor_split_finalize()

```
static void gst_tensor_split_finalize (
            GObject * object )  [static]
```

finalize function for tensor split (gst element vmethod)

Definition at line 202 of file gsttensor_split.c.

Here is the call graph for this function:



Here is the caller graph for this function:

9.62.4.10 gst_tensor_split_get_capsparam()

```
static gboolean gst_tensor_split_get_capsparam (
            GstTensorSplit * split,
            GstCaps * caps )  [static]
```

Set Caps in pad.

Parameters

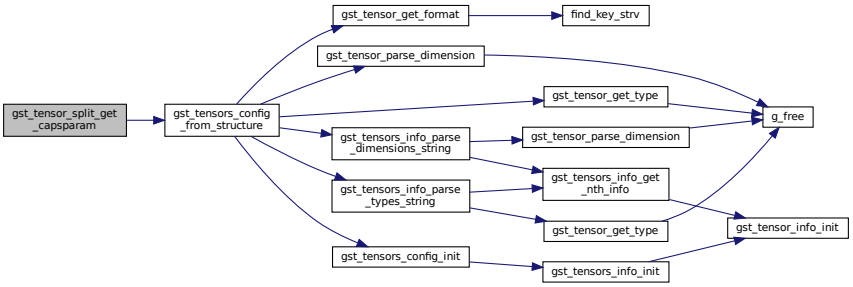
split	GstTensorSplit object
caps	incoming capability

Returns

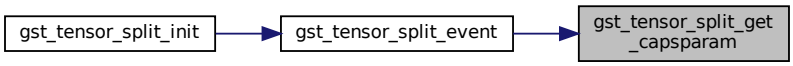
TRUE/FALSE (if successfully generate & set cap, return TRUE)

Definition at line 220 of file gsttensor_split.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.62.4.11 gst_tensor_split_get_property()

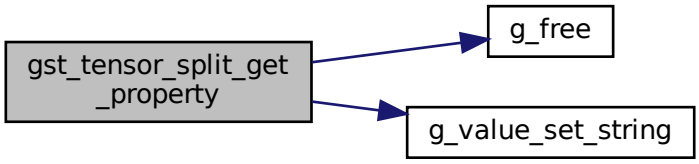
```
static void gst_tensor_split_get_property (
            GObject * object,
            guint prop_id,
            GValue * value,
            GParamSpec * pspec )  [static]
```

Get property (gst element vmethod)

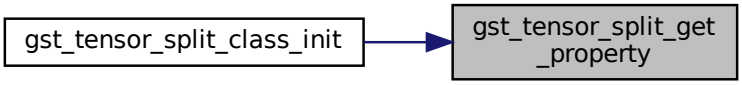
If i = 1, this is previous p. Otherwise, it's previous g_strjoin result.

Definition at line 653 of file gsttensor_split.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.62.4.12 gst_tensor_split_get_splitted()

```
static GstMemory* gst_tensor_split_get_splitted (
            GstTensorSplit * split,
            GstBuffer * buffer,
            gint nth )  [static]
```

Make splitted tensor.

Parameters

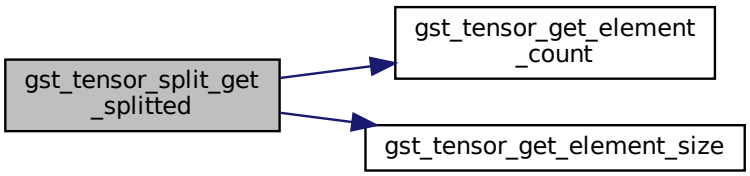
split	TensorSplit object
buffer	gstbuffer form src
nth	orther of tensor

Returns

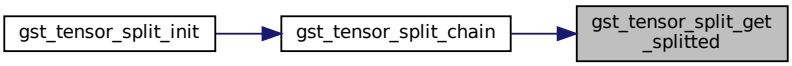
return GstMemory for splitted tensor

Definition at line 415 of file gsttensor_split.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.62.4.13 gst_tensor_split_get_tensor_pad()

```
static GstTensorPad* gst_tensor_split_get_tensor_pad (
            GstTensorSplit * split,
            GstBuffer * inbuf,
            gboolean * created,
            guint nth )  [static]
```

Checking if the source pad is created and if not, create TensorPad.

Parameters

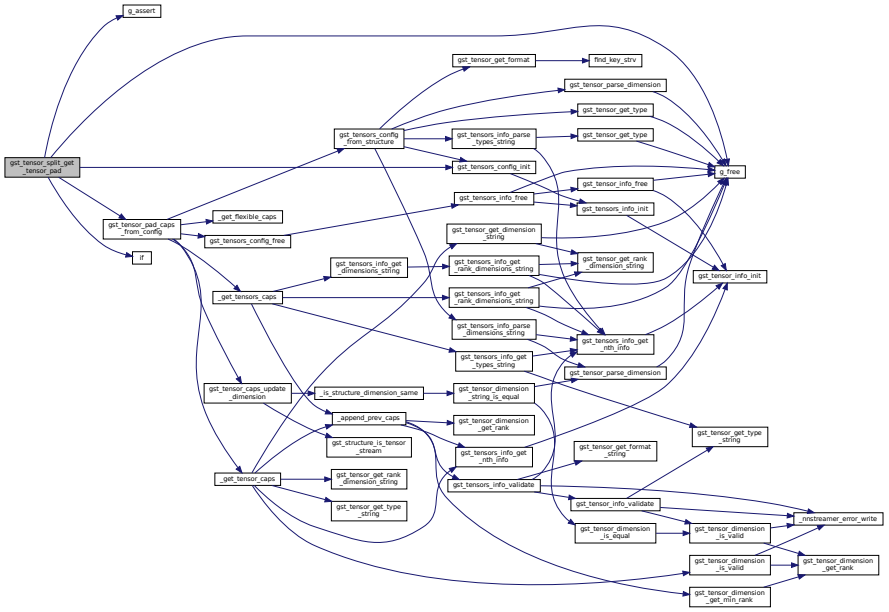
	split	TensorSplit Object
	inbuf	inputbuf GstBuffer Object including GstMeta
out	*created*	will be updated in this function
	nth	source ordering

Returns

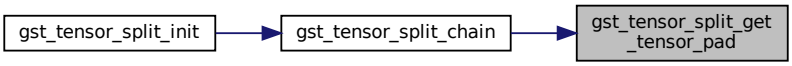
TensorPad if pad is already created, then return created pad. If not return new pad after creation.

Definition at line 276 of file gsttensor_split.c.

Here is the call graph for this function:



Here is the caller graph for this function:



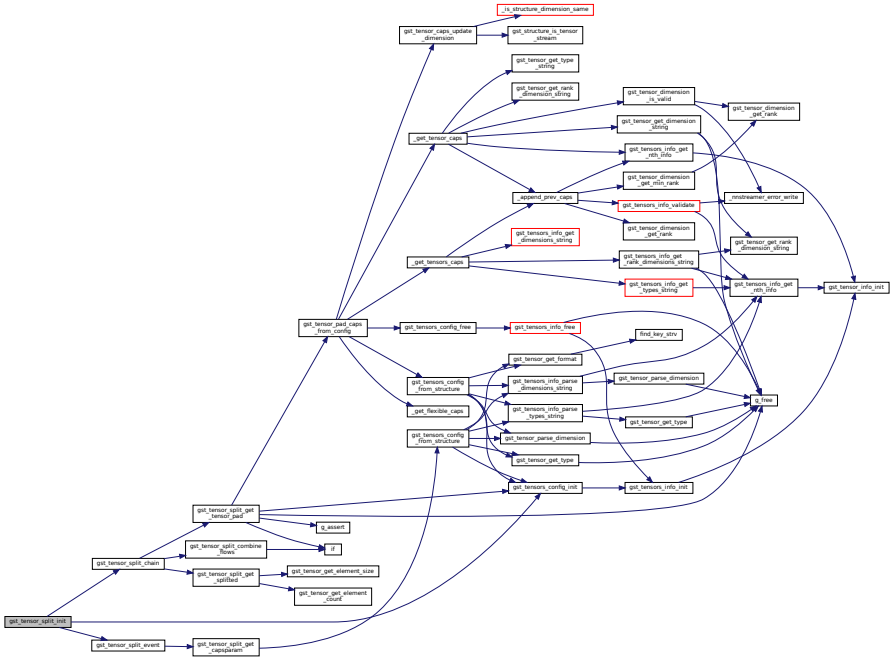
9.62.4.14 gst_tensor_split_init()

```
static void gst_tensor_split_init (
            GstTensorSplit * split )  [static]
```

initialize the new element instantiate pads and add them to element set pad callback functions initialize instance structure

Definition at line 160 of file gsttensor_split.c.

Here is the call graph for this function:



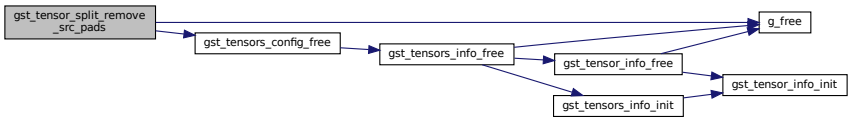
9.62.4.15 gst_tensor_split_remove_src_pads()

```
static void gst_tensor_split_remove_src_pads (
            GstTensorSplit * split )  [static]
```

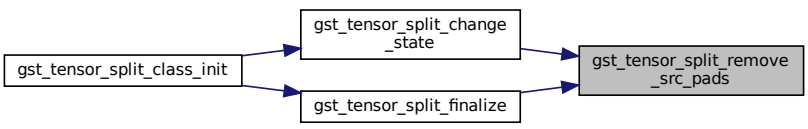
function to remove srcpad list

Definition at line 184 of file gsttensor_split.c.

Here is the call graph for this function:



Here is the caller graph for this function:



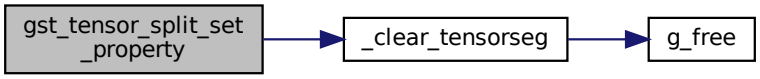
9.62.4.16 gst_tensor_split_set_property()

```
static void gst_tensor_split_set_property (
            GObject * object,
            guint prop_id,
            const GValue * value,
            GParamSpec * pspec )  [static]
```

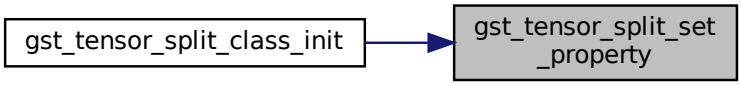
Get property (gst element vmethod)

Definition at line 578 of file gsttensor_split.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.62.5 Variable Documentation

9.62.5.1 sink_templ

```
GstStaticPadTemplate sink_templ  [static]
```

Initial value:

```
= GST_STATIC_PAD_TEMPLATE ("sink",
    GST_PAD_SINK,
    GST_PAD_ALWAYS,
    GST_STATIC_CAPS (CAPS_STRING))
```

Definition at line 84 of file gsttensor_split.c.

9.62.5.2 src_templ

```
GstStaticPadTemplate src_templ  [static]
```

Initial value:

```
= GST_STATIC_PAD_TEMPLATE ("src_%u",
    GST_PAD_SRC,
    GST_PAD_SOMETIMES,
    GST_STATIC_CAPS (CAPS_STRING))
```

the capabilities of the inputs and outputs. describe the real formats here.

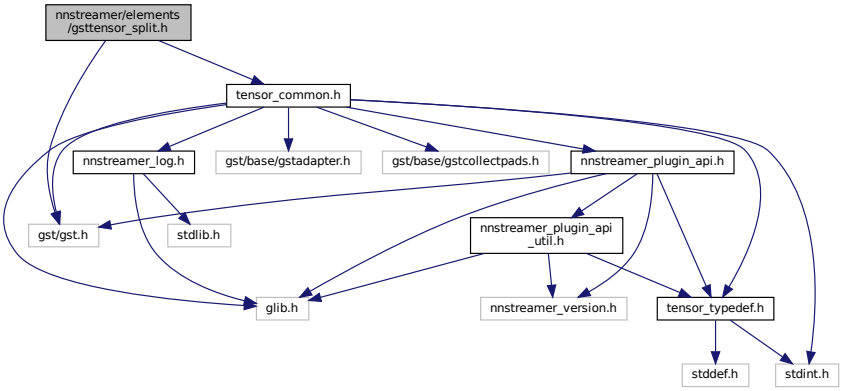
Definition at line 79 of file gsttensor_split.c.

9.63 nnstreamer/elements/gsttensor_split.h File Reference

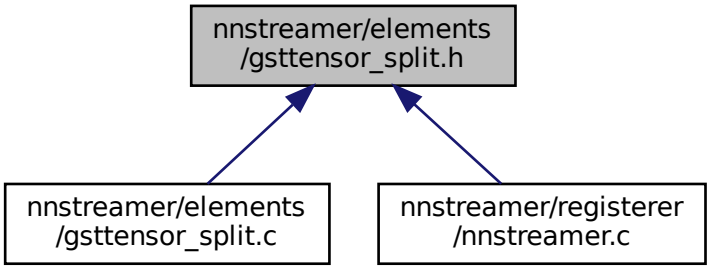
GStreamer plugin to split tensor (as a filter for other general neural network filters)

```
#include <gst/gst.h>
#include <tensor_common.h>
```

Include dependency graph for gsttensor_split.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct _GstTensorSplit

  Tensor Split data structure.
- struct _GstTensorSplitClass

  GstTensorSplitClass inherits GstElementClass.

Macros

- #define GST_TYPE_TENSOR_SPLIT (gst_tensor_split_get_type ())
- #define GST_TENSOR_SPLIT(obj) (G_TYPE_CHECK_INSTANCE_CAST ((obj), GST_TYPE_TENSOR_SPLIT, GstTensorSplit))
- #define GST_TENSOR_SPLIT_CLASS(klass) (G_TYPE_CHECK_CLASS_CAST ((klass), GST_TYPE_TENSOR_SPLIT, GstTensorSplitClass))
- #define GST_TENSOR_SPLIT_GET_CLASS(obj) (G_TYPE_INSTANCE_GET_CLASS ((obj), GST_TYPE_TENSOR_SPLIT, GstTensorSplitClass))
- #define GST_IS_TENSOR_SPLIT(obj) (G_TYPE_CHECK_INSTANCE_TYPE((obj),GST_TYPE_TENSOR_SPLIT))
- #define GST_IS_TENSOR_SPLIT_CLASS(klass) (G_TYPE_CHECK_CLASS_TYPE((klass),GST_TYPE_TENSOR_SPLIT))
- #define GST_TENSOR_SPLIT_CAST(obj) ((GstTensorSplit*)(obj))

Typedefs

- typedef struct _GstTensorSplit GstTensorSplit
- typedef struct _GstTensorSplitClass GstTensorSplitClass

Functions

- GType gst_tensor_split_get_type (void)

  Get Type function required for gst elements.

9.63.1 Detailed Description

GStreamer plugin to split tensor (as a filter for other general neural network filters)

GStreamer Copyright (C) 2005 Thomas Vander Stichele thomas@apestaart.org Copyright (C) 2005 Ronald S. Bultje rbultje@ronald.bitfreak.net Copyright (C) 2018 Jijoong Moon jijoong.moon@samsung.com

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Library General Public License as published by the Free Software Foundation; version 2.1 of the License.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Library General Public License for more details.

Date

27 Aug 2018

See also

https://github.com/nnstreamer/nnstreamer

Author

Jijoong Moon jijoong.moon@samsung.com

Bug No known bugs except for NYI items

9.63.2 Macro Definition Documentation

9.63.2.1 GST_IS_TENSOR_SPLIT

```
#define GST_IS_TENSOR_SPLIT(
              obj ) (G_TYPE_CHECK_INSTANCE_TYPE((obj),GST_TYPE_TENSOR_SPLIT))
```

Definition at line 40 of file gsttensor_split.h.

9.63.2.2 GST_IS_TENSOR_SPLIT_CLASS

```
#define GST_IS_TENSOR_SPLIT_CLASS(
              klass ) (G_TYPE_CHECK_CLASS_TYPE((klass),GST_TYPE_TENSOR_SPLIT))
```

Definition at line 41 of file gsttensor_split.h.

9.63.2.3 GST_TENSOR_SPLIT

```
#define GST_TENSOR_SPLIT(
              obj ) (G_TYPE_CHECK_INSTANCE_CAST ((obj), GST_TYPE_TENSOR_SPLIT, GstTensorSplit))
```

Definition at line 37 of file gsttensor_split.h.

9.63.2.4 GST_TENSOR_SPLIT_CAST

```
#define GST_TENSOR_SPLIT_CAST(
              obj ) ((GstTensorSplit*)(obj))
```

Definition at line 42 of file gsttensor_split.h.

9.63.2.5 GST_TENSOR_SPLIT_CLASS

```
#define GST_TENSOR_SPLIT_CLASS(
              klass ) (G_TYPE_CHECK_CLASS_CAST ((klass), GST_TYPE_TENSOR_SPLIT, GstTensorSplitClass))
```

Definition at line 38 of file gsttensor_split.h.

9.63.2.6 GST_TENSOR_SPLIT_GET_CLASS

```
#define GST_TENSOR_SPLIT_GET_CLASS(
              obj ) (G_TYPE_INSTANCE_GET_CLASS ((obj), GST_TYPE_TENSOR_SPLIT, GstTensorSplitClass))
```

Definition at line 39 of file gsttensor_split.h.

9.63.2.7 GST_TYPE_TENSOR_SPLIT

```
#define GST_TYPE_TENSOR_SPLIT (gst_tensor_split_get_type ())
```

Definition at line 36 of file gsttensor_split.h.

9.63.3 Typedef Documentation

9.63.3.1 GstTensorSplit

```
typedef struct _GstTensorSplit GstTensorSplit
```

Definition at line 44 of file gsttensor_split.h.

9.63.3.2 GstTensorSplitClass

```
typedef struct _GstTensorSplitClass GstTensorSplitClass
```

Definition at line 45 of file gsttensor_split.h.

9.63.4 Function Documentation

9.63.4.1 gst_tensor_split_get_type()

```
GType gst_tensor_split_get_type (
             void  )
```

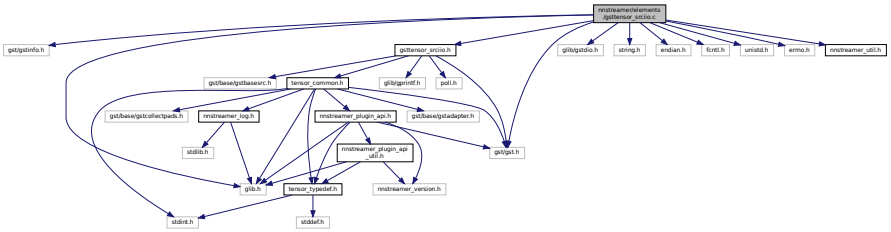
Get Type function required for gst elements.

9.64 nnstreamer/elements/gsttensor_srciio.c File Reference

GStreamer plugin to capture sensor data as tensor(s)

```
#include <gst/gstinfo.h>
#include <gst/gst.h>
#include <glib.h>
#include <glib/gstdio.h>
#include <string.h>
#include <endian.h>
#include <fcntl.h>
#include <unistd.h>
#include <errno.h>
#include <nnstreamer_util.h>
#include "gsttensor_srciio.h"
```

Include dependency graph for gsttensor_srciio.c:



Macros

- #define DBG (!self->silent)

  Macro for debug mode.
- #define GST_CAT_DEFAULT gst_tensor_src_iio_debug
- #define PROCESS_SCANNED_DATA(DTYPE_UNSIGNED, DTYPE_SIGNED)

  Macro to generate data processing functions for various types.
- #define DEFAULT_PROP_BASE_DIRECTORY "/sys/bus/iio/devices"

  IIO system paths.
- #define DEFAULT_PROP_DEV_DIRECTORY "/dev"
- #define CHANNELS_ENABLED_AUTO_CHAR "auto"

  iio device channel enabled mode
- #define CHANNELS_ENABLED_ALL_CHAR "all"
- #define DEFAULT_OPERATING_CHANNELS_ENABLED CHANNELS_ENABLED_AUTO_CHAR
- #define MODE_ONE_SHOT "one-shot"

  tensor_src_iio device modes
- #define MODE_CONTINUOUS "continuous"
- #define DEFAULT_OPERATING_MODE MODE_CONTINUOUS
- #define DEFAULT_PROP_SILENT TRUE

  Flag to print minimized log.
- #define DEFAULT_PROP_STRING NULL

  Flag for general default value of string.
- #define MIN_BUFFER_CAPACITY 1

  Minimum and maximum buffer length for iio.
- #define MAX_BUFFER_CAPACITY G_MAXUINT
- #define DEFAULT_BUFFER_CAPACITY 1

- #define MIN_FREQUENCY 0

  Minimum and maximum operating frequency for the device Frequency 0 chooses the first available frequency supported by device.
- #define MAX_FREQUENCY G_MAXULONG
- #define DEFAULT_FREQUENCY 0
- #define MIN_POLL_TIMEOUT -1

  Minimum and maximum polling timeout for the buffered reading.
- #define MAX_POLL_TIMEOUT G_MAXINT
- #define DEFAULT_POLL_TIMEOUT 10000
- #define DEFAULT_MERGE_CHANNELS TRUE

  Default behavior on merging channels.
- #define DEFAULT_PROP_DEVICE_NUM -1

  default trigger and device numbers
- #define DEFAULT_PROP_TRIGGER_NUM -1
- #define BLOCKSIZE 1
- #define DEVICE "device"

  IIO devices/triggers.
- #define BUFFER "buffer"
- #define TRIGGER "trigger"
- #define CHANNELS "scan_elements"
- #define IIO "iio:"
- #define TIMESTAMP "timestamp"
- #define DEVICE_PREFIX IIO DEVICE
- #define TRIGGER_PREFIX IIO TRIGGER
- #define CURRENT_TRIGGER "current_trigger"
- #define EN_SUFFIX "_en"

  IIO device channels.
- #define INDEX_SUFFIX "_index"
- #define TYPE_SUFFIX "_type"
- #define SCALE_SUFFIX "_scale"
- #define OFFSET_SUFFIX "_offset"
- #define NAME_FILE "name"

  filenames for IIO devices/triggers characteristics
- #define AVAIL_FREQUENCY_FILE "sampling_frequency_available"
- #define SAMPLING_FREQUENCY "sampling_frequency"
- #define gst_tensor_src_iio_parent_class parent_class

Enumerations

- enum {
  PROP_0, PROP_MODE, PROP_SILENT, PROP_BASE_DIRECTORY,
  PROP_DEV_DIRECTORY, PROP_DEVICE, PROP_DEVICE_NUM, PROP_TRIGGER,
  PROP_TRIGGER_NUM, PROP_CHANNELS, PROP_BUFFER_CAPACITY, PROP_FREQUENCY,
  PROP_MERGE_CHANNELS, PROP_POLL_TIMEOUT }

  tensor_src_iio properties.

Functions

- GST_DEBUG_CATEGORY_STATIC (gst_tensor_src_iio_debug)
- g_assert (sizeof(DTYPE_UNSIGNED)==sizeof(DTYPE_SIGNED))
- value_unsigned & if (prop->is_signed)
- PROCESS_SCANNED_DATA (guint8, gint8)
- PROCESS_SCANNED_DATA (guint16, gint16)
- PROCESS_SCANNED_DATA (guint32, gint32)
- PROCESS_SCANNED_DATA (guint64, gint64)
- static void gst_tensor_src_iio_set_property (GObject *object, guint prop_id, const GValue *value, GParamSpec *pspec)

  set tensor_src_iio properties
- static void gst_tensor_src_iio_get_property (GObject *object, guint prop_id, GValue *value, GParamSpec *pspec)

  get tensor_src_iio properties
- static void gst_tensor_src_iio_finalize (GObject *object)

  finalize the instance
- static gboolean gst_tensor_src_iio_start (GstBaseSrc *src)

  start function, called when state changed null to ready. load the device and init the device resources
- static gboolean gst_tensor_src_iio_stop (GstBaseSrc *src)

  stop function, called when state changed ready to null.
- static GstStateChangeReturn gst_tensor_src_iio_change_state (GstElement *element, GstStateChange transition)

  Perform state change.
- static gboolean gst_tensor_src_iio_event (GstBaseSrc *src, GstEvent *event)

  handle events
- static gboolean gst_tensor_src_iio_set_caps (GstBaseSrc *src, GstCaps *caps)

  set new caps
- static GstCaps * gst_tensor_src_iio_get_caps (GstBaseSrc *src, GstCaps *filter)

  get caps of subclass
- static GstCaps * gst_tensor_src_iio_fixate (GstBaseSrc *src, GstCaps *caps)

  fixate the caps when needed during negotiation
- static gboolean gst_tensor_src_iio_is_seekable (GstBaseSrc *src)

  check if source supports seeking
- static GstFlowReturn gst_tensor_src_iio_create (GstBaseSrc *src, guint64 offset, guint size, GstBuffer **buffer)

  create a buffer with requested size and offset
- static GstFlowReturn gst_tensor_src_iio_fill (GstBaseSrc *src, guint64 offset, guint size, GstBuffer *buffer)

  fill the buffer with data
- static void gst_tensor_src_iio_get_times (GstBaseSrc *basesrc, GstBuffer *buffer, GstClockTime *start, GstClockTime *end)

  returns the time for the buffers
- G_DEFINE_TYPE (GstTensorSrcIIO, gst_tensor_src_iio, GST_TYPE_BASE_SRC)
- static void gst_tensor_src_iio_class_init (GstTensorSrcIIOClass *klass)

  initialize the tensor_src_iio class.
- static void gst_tensor_src_iio_channel_properties_free (gpointer data)

  delete GstTensorSrcIIODeviceProperties structure
- static void gst_tensor_src_iio_device_properties_init (GstTensorSrcIIODeviceProperties *prop)

  initialize GstTensorSrcIIODeviceProperties structure
- static void gst_tensor_src_iio_init (GstTensorSrcIIO *self)

  initialize tensor_src_iio element.
- static gint gst_tensor_src_merge_tensor_by_type (GstTensorInfo *info, guint size, guint dir)

merge multiple other/tensor

- static gint gst_tensor_src_iio_get_id_by_name (const gchar *dir_name, const gchar *name, const gchar *prefix)

  check if device/trigger with the given name exists
- static gchar * gst_tensor_src_iio_get_name_by_id (const gchar *dir_name, const gint id, const gchar *prefix)

  check if device/trigger with the given id exists
- static gboolean gst_tensor_src_iio_get_float_from_file (const gchar *dirname, const gchar *name, const gchar *suffix, gfloat *value)

  parse float value from the file
- static gboolean gst_tensor_src_iio_set_channel_type (GstTensorSrcIIOChannelProperties *prop, const gchar *contents)

  get type info about the channel from the string
- static gchar * gst_tensor_src_iio_get_generic_name (const gchar *channel_name)

  get generic name for channel from the string
- static gint gst_tensor_channel_list_sort_cmp (gconstpointer a, gconstpointer b)

  compare channels for sort based on their indices
- static void gst_tensor_channel_list_filter_enabled (gpointer data, gpointer user_data)

  compare channels for filtering if enabled
- static gint gst_tensor_src_iio_get_all_channel_info (GstTensorSrcIIO *self, const gchar *dir_name)

  get info about all the channels in the device
- static gint64 gst_tensor_src_iio_get_available_frequency (const gchar *base_dir, const guint64 frequency)

  return sampling frequency given the frequency input from user
- static gboolean gst_tensor_write_sysfs_string (GstTensorSrcIIO *self, const gchar *file, const gchar *base_dir, const gchar *contents)

  write the string in to the file
- static gboolean gst_tensor_write_sysfs_int (GstTensorSrcIIO *self, const gchar *file, const gchar *base_dir, const gint contents)

  write the int in to the file
- static gboolean gst_tensor_set_all_channels (GstTensorSrcIIO *self, const gint contents)

  set value to all the channels
- static guint gst_tensor_get_size_from_channels (GList *channels)

  get the size of the combined data from channels
- static gboolean gst_tensor_src_iio_create_config (GstTensorSrcIIO *tensor_src_iio)

  create the structure for the caps to update the src pad caps
- static gboolean gst_tensor_src_iio_setup_device_properties (GstTensorSrcIIO *self)

  setup device using name/id
- static gboolean gst_tensor_src_iio_setup_trigger_properties (GstTensorSrcIIO *self)

  setup trigger using name/id
- static gboolean gst_tensor_src_iio_setup_sampling_frequency (GstTensorSrcIIO *self)

  setup device sampling frequency
- static gboolean gst_tensor_src_iio_setup_scan_channels (GstTensorSrcIIO *self)

  setup scan channels for the device
- static gboolean gst_tensor_src_iio_setup_device_buffer (GstTensorSrcIIO *self)

  setup device using name/id
- static void gst_tensor_src_restore_iio_device (GstTensorSrcIIO *self)

  restore the iio device to its original device.
- static gboolean gst_tensor_src_iio_process_scanned_data (GstTensorSrcIIOChannelProperties *prop, gchar *data, gfloat *buffer_map)

  process the scanned data from IIO device

Variables

- GstTensorSrcIIOChannelProperties * prop

  DTYPE_UNSIGNED (.
- GstTensorSrcIIOChannelProperties DTYPE_UNSIGNED value_unsigned
- else
- return value_float

9.64.1 Detailed Description

GStreamer plugin to capture sensor data as tensor(s)

GStreamer Tensor_Source_IIO Copyright (C) 2005 Thomas Vander Stichele thomas@apestaart.org Copyright (C) 2005 Ronald S. Bultje rbultje@ronald.bitfreak.net Copyright (C) 2019 Parichay Kapoor pk.kapoor@samsung.com

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Library General Public License as published by the Free Software Foundation; version 2.1 of the License.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Library General Public License for more details.

Date

27 Feb 2019

See also

http://github.com/nnstreamer/nnstreamer

Author

Parichay Kapoor pk.kapoor@samsung.com

Bug No known bugs except for NYI items

Todo support specific channels as input

handle timestamp received from device

This is the plugin to capture data from sensors and convert them to tensor format. Current implementation will support accelerators, light and gyro sensors.

9.64.2 Macro Definition Documentation

9.64.2.1 AVAIL_FREQUENCY_FILE

```
#define AVAIL_FREQUENCY_FILE "sampling_frequency_available"
```

Definition at line 244 of file gsttensor_srciio.c.

9.64.2.2 BLOCKSIZE

```
#define BLOCKSIZE 1
```

blocksize for buffer

Definition at line 216 of file gsttensor_srciio.c.

9.64.2.3 BUFFER

```
#define BUFFER "buffer"
```

Definition at line 222 of file gsttensor_srciio.c.

9.64.2.4 CHANNELS

```
#define CHANNELS "scan_elements"
```

Definition at line 224 of file gsttensor_srciio.c.

9.64.2.5 CHANNELS_ENABLED_ALL_CHAR

```
#define CHANNELS_ENABLED_ALL_CHAR "all"
```

Definition at line 160 of file gsttensor_srciio.c.

9.64.2.6 CHANNELS_ENABLED_AUTO_CHAR

```
#define CHANNELS_ENABLED_AUTO_CHAR "auto"
```

iio device channel enabled mode

Definition at line 159 of file gsttensor_srciio.c.

9.64.2.7 CURRENT_TRIGGER

```
#define CURRENT_TRIGGER "current_trigger"
```

Definition at line 229 of file gsttensor_srciio.c.

9.64.2.8 DBG

```
#define DBG (!self->silent)
```

Macro for debug mode.

SECTION:element-tensor_src_iio

#tensor_src_iio extends #gstbasesrc source element to handle Linux Industrial I/O sensors as input. IIO sources are only supported in buffered mode. Source elements only support push mode scheduling as a live source.

#tensor_src_iio supports configuring the device as well as the trigger via properties. Buffer capacity, frequency and scan channels to be read can be configured before PLAYING the stream. The configuration is supported only in states <= READY. Setting the state back to NULL restores the original configuration of the IIO device. The source can be configured to work with trigger for the source or read the data from the device at regular time intervals. Device name/number is the only necessary configuration needed to run the element (other configuration parameters is optional).

The output caps is either of other/tensor or other/tensors.

Data from various channels can be merged to form 1 other/tensor. Final caps of the src pad is of the following format: <itemizedlist> <listitem>

Dimension 0 : Channel number</listitem>

<listitem>

Dimension 1 : buffer capacity</listitem>

</itemizedlist> Other dimensions are not utilized. The data in the dimension 0 is sorted on the basis of the indexing of the channels provided by the IIO device.

The enabling of buffer for data capture is performed when transitioning from PAUSED to PLAYING state. This leads to automated synchronization handled by gstreamer. Buffer duration and timestamps set by #gstbasesrc remain in sync with linux IIO timestamps.

<refsect2> <title>Example launch line</title> |[gst-launch -v -m tensor_src_iio device-number=0 ! fakesink]| </refsect2>

Definition at line 97 of file gsttensor_srciio.c.

9.64.2.9 DEFAULT_BUFFER_CAPACITY

```
#define DEFAULT_BUFFER_CAPACITY 1
```

Definition at line 185 of file gsttensor_srciio.c.

9.64.2.10 DEFAULT_FREQUENCY

```
#define DEFAULT_FREQUENCY 0
```

Definition at line 193 of file gsttensor_srciio.c.

9.64.2.11 DEFAULT_MERGE_CHANNELS

```
#define DEFAULT_MERGE_CHANNELS TRUE
```

Default behavior on merging channels.

Definition at line 205 of file gsttensor_srciio.c.

9.64.2.12 DEFAULT_OPERATING_CHANNELS_ENABLED

```
#define DEFAULT_OPERATING_CHANNELS_ENABLED CHANNELS_ENABLED_AUTO_CHAR
```

Definition at line 161 of file gsttensor_srciio.c.

9.64.2.13 DEFAULT_OPERATING_MODE

```
#define DEFAULT_OPERATING_MODE MODE_CONTINUOUS
```

Definition at line 168 of file gsttensor_srciio.c.

9.64.2.14 DEFAULT_POLL_TIMEOUT

```
#define DEFAULT_POLL_TIMEOUT 10000
```

Definition at line 200 of file gsttensor_srciio.c.

9.64.2.15 DEFAULT_PROP_BASE_DIRECTORY

```
#define DEFAULT_PROP_BASE_DIRECTORY "/sys/bus/iio/devices"
```

IIO system paths.

Definition at line 153 of file gsttensor_srciio.c.

9.64.2.16 DEFAULT_PROP_DEV_DIRECTORY

```
#define DEFAULT_PROP_DEV_DIRECTORY "/dev"
```

Definition at line 154 of file gsttensor_srciio.c.

9.64.2.17 DEFAULT_PROP_DEVICE_NUM

```
#define DEFAULT_PROP_DEVICE_NUM -1
```

default trigger and device numbers

Definition at line 210 of file gsttensor_srciio.c.

9.64.2.18 DEFAULT_PROP_SILENT

```
#define DEFAULT_PROP_SILENT TRUE
```

Flag to print minimized log.

Definition at line 173 of file gsttensor_srciio.c.

9.64.2.19 DEFAULT_PROP_STRING

```
#define DEFAULT_PROP_STRING NULL
```

Flag for general default value of string.

Definition at line 178 of file gsttensor_srciio.c.

9.64.2.20 DEFAULT_PROP_TRIGGER_NUM

```
#define DEFAULT_PROP_TRIGGER_NUM -1
```

Definition at line 211 of file gsttensor_srciio.c.

9.64.2.21 DEVICE

```
#define DEVICE "device"
```

IIO devices/triggers.

Definition at line 221 of file gsttensor_srciio.c.

9.64.2.22 DEVICE_PREFIX

```
#define DEVICE_PREFIX IIO DEVICE
```

Definition at line 227 of file gsttensor_srciio.c.

9.64.2.23 EN_SUFFIX

```
#define EN_SUFFIX "_en"
```

IIO device channels.

Definition at line 234 of file gsttensor_srciio.c.

9.64.2.24 GST_CAT_DEFAULT

```
#define GST_CAT_DEFAULT gst_tensor_src_iio_debug
```

Definition at line 101 of file gsttensor_srciio.c.

9.64.2.25 gst_tensor_src_iio_parent_class

```
#define gst_tensor_src_iio_parent_class parent_class
```

internal functions

Definition at line 280 of file gsttensor_srciio.c.

9.64.2.26 IIO

```
#define IIO "iio:"
```

Definition at line 225 of file gsttensor_srciio.c.

9.64.2.27 INDEX_SUFFIX

```
#define INDEX_SUFFIX "_index"
```

Definition at line 235 of file gsttensor_srciio.c.

9.64.2.28 MAX_BUFFER_CAPACITY

```
#define MAX_BUFFER_CAPACITY G_MAXUINT
```

Definition at line 184 of file gsttensor_srciio.c.

9.64.2.29 MAX_FREQUENCY

```
#define MAX_FREQUENCY G_MAXULONG
```

Definition at line 192 of file gsttensor_srciio.c.

9.64.2.30 MAX_POLL_TIMEOUT

```
#define MAX_POLL_TIMEOUT G_MAXINT
```

Definition at line 199 of file gsttensor_srciio.c.

9.64.2.31 MIN_BUFFER_CAPACITY

```
#define MIN_BUFFER_CAPACITY 1
```

Minimum and maximum buffer length for iio.

Definition at line 183 of file gsttensor_srciio.c.

9.64.2.32 MIN_FREQUENCY

```
#define MIN_FREQUENCY 0
```

Minimum and maximum operating frequency for the device Frequency 0 chooses the first available frequency supported by device.

Definition at line 191 of file gsttensor_srciio.c.

9.64.2.33 MIN_POLL_TIMEOUT

```
#define MIN_POLL_TIMEOUT -1
```

Minimum and maximum polling timeout for the buffered reading.

Definition at line 198 of file gsttensor_srciio.c.

9.64.2.34 MODE_CONTINUOUS

```
#define MODE_CONTINUOUS "continuous"
```

Definition at line 167 of file gsttensor_srciio.c.

9.64.2.35 MODE_ONE_SHOT

```
#define MODE_ONE_SHOT "one-shot"
```

tensor_src_iio device modes

Definition at line 166 of file gsttensor_srciio.c.

9.64.2.36 NAME_FILE

```
#define NAME_FILE "name"
```

filenames for IIO devices/triggers characteristics

Definition at line 243 of file gsttensor_srciio.c.

9.64.2.37 OFFSET_SUFFIX

```
#define OFFSET_SUFFIX "_offset"
```

Definition at line 238 of file gsttensor_srciio.c.

9.64.2.38 PROCESS_SCANNED_DATA

```
#define PROCESS_SCANNED_DATA(
              DTYPE_UNSIGNED,
              DTYPE_SIGNED )
```

Value:

```
 \
static gfloat \
gst_tensor_src_iio_process_scanned_data_from_
```

Macro to generate data processing functions for various types.

Definition at line 106 of file gsttensor_srciio.c.

9.64.2.39 SAMPLING_FREQUENCY

```
#define SAMPLING_FREQUENCY "sampling_frequency"
```

Definition at line 245 of file gsttensor_srciio.c.

9.64.2.40 SCALE_SUFFIX

```
#define SCALE_SUFFIX "_scale"
```

Definition at line 237 of file gsttensor_srciio.c.

9.64.2.41 TIMESTAMP

```
#define TIMESTAMP "timestamp"
```

Definition at line 226 of file gsttensor_srciio.c.

9.64.2.42 TRIGGER

```
#define TRIGGER "trigger"
```

Definition at line 223 of file gsttensor_srciio.c.

9.64.2.43 TRIGGER_PREFIX

```
#define TRIGGER_PREFIX IIO TRIGGER
```

Definition at line 228 of file gsttensor_srciio.c.

9.64.2.44 TYPE_SUFFIX

```
#define TYPE_SUFFIX "_type"
```

Definition at line 236 of file gsttensor_srciio.c.

9.64.3 Enumeration Type Documentation

9.64.3.1 anonymous enum

```
anonymous enum
```

tensor_src_iio properties.

Enumerator

PROP_0	
PROP_MODE	
PROP_SILENT	
PROP_BASE_DIRECTORY	
PROP_DEV_DIRECTORY	
PROP_DEVICE	
PROP_DEVICE_NUM	
PROP_TRIGGER	
PROP_TRIGGER_NUM	
PROP_CHANNELS	
PROP_BUFFER_CAPACITY	
PROP_FREQUENCY	
PROP_MERGE_CHANNELS	
PROP_POLL_TIMEOUT	

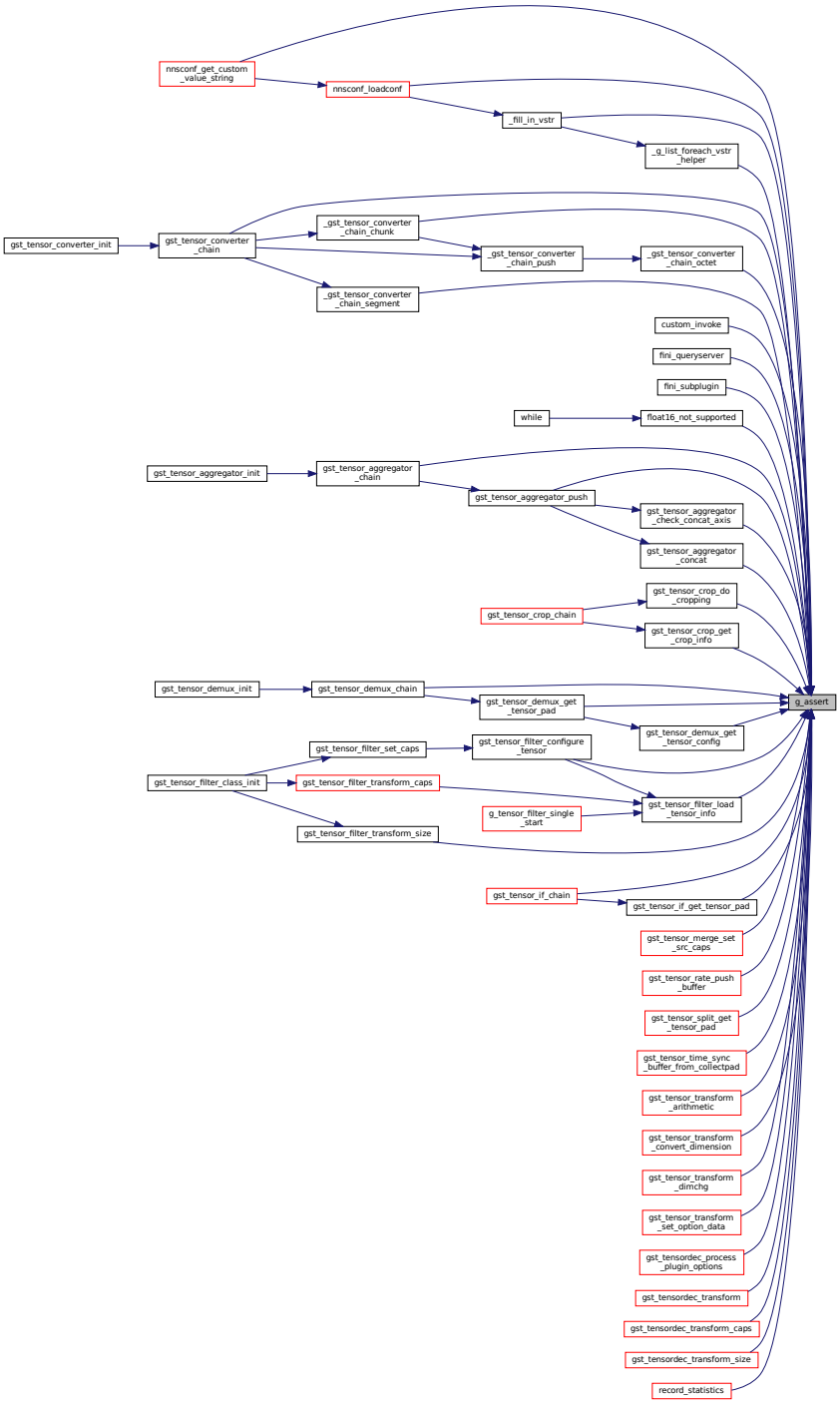
Definition at line 132 of file gsttensor_srciio.c.

9.64.4 Function Documentation

9.64.4.1 g_assert()

```
g_assert (
            sizeof(DTYPE_UNSIGNED)  = =sizeof(DTYPE_SIGNED) )
```

Here is the caller graph for this function:



9.64.4.2 G_DEFINE_TYPE()

```
G_DEFINE_TYPE (
            GstTensorSrcIIO ,
```

```
            gst_tensor_src_iio ,
            GST_TYPE_BASE_SRC  )
```

9.64.4.3 GST_DEBUG_CATEGORY_STATIC()

```
GST_DEBUG_CATEGORY_STATIC (
            gst_tensor_src_iio_debug  )
```

9.64.4.4 gst_tensor_channel_list_filter_enabled()

```
static void gst_tensor_channel_list_filter_enabled (
            gpointer data,
            gpointer user_data )  [static]
```

compare channels for filtering if enabled

Parameters

in	*data*	Pointer of the data of the element
	[in/out]	user_data Pointer to the address of the list to be filtered

Definition at line 836 of file gsttensor_srciio.c.

9.64.4.5 gst_tensor_channel_list_sort_cmp()

```
static gint gst_tensor_channel_list_sort_cmp (
            gconstpointer a,
            gconstpointer b )  [static]
```

compare channels for sort based on their indices

Parameters

in	*a*	First param to be compared
in	*b*	Second param to be compared

Returns

negative if a**b**

Definition at line 822 of file gsttensor_srciio.c.

9.64.4.6 gst_tensor_get_size_from_channels()

```
static guint gst_tensor_get_size_from_channels (
            GList * channels )  [static]
```

get the size of the combined data from channels

Parameters

in	*channels*	List of all the channels

Returns

Size of one scan of data combined from all the channels

Also evaluates the location of each channel in the buffer

Definition at line 1499 of file gsttensor_srciio.c.

9.64.4.7 gst_tensor_set_all_channels()

```
static gboolean gst_tensor_set_all_channels (
            GstTensorSrcIIO * self,
            const gint contents )  [static]
```

set value to all the channels

Parameters

in	*self*	Tensor src IIO object
in	*contents*	Data to be written to the file

Returns

True if write was successful, false if failure on any channel

Definition at line 1469 of file gsttensor_srciio.c.

9.64.4.8 gst_tensor_src_iio_change_state()

```
static GstStateChangeReturn gst_tensor_src_iio_change_state (
            GstElement * element,
            GstStateChange transition )  [static]
```

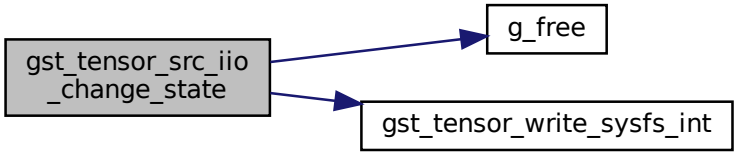
Perform state change.

enable the buffer for the data to be captured

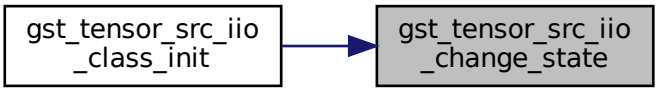
disable the buffer

Definition at line 2228 of file gsttensor_srciio.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.64.4.9 gst_tensor_src_iio_channel_properties_free()

```
static void gst_tensor_src_iio_channel_properties_free (
             gpointer data )  [static]
```

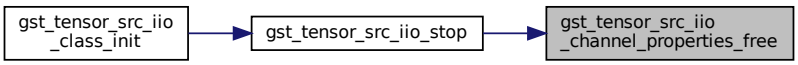
delete GstTensorSrcIIODeviceProperties structure

Parameters

in	*data*	Data pointer to be freed

Definition at line 411 of file gsttensor_srciio.c.

Here is the caller graph for this function:



9.64.4.10 gst_tensor_src_iio_class_init()

```
static void gst_tensor_src_iio_class_init (
            GstTensorSrcIIOClass * klass )  [static]
```

initialize the tensor_src_iio class.

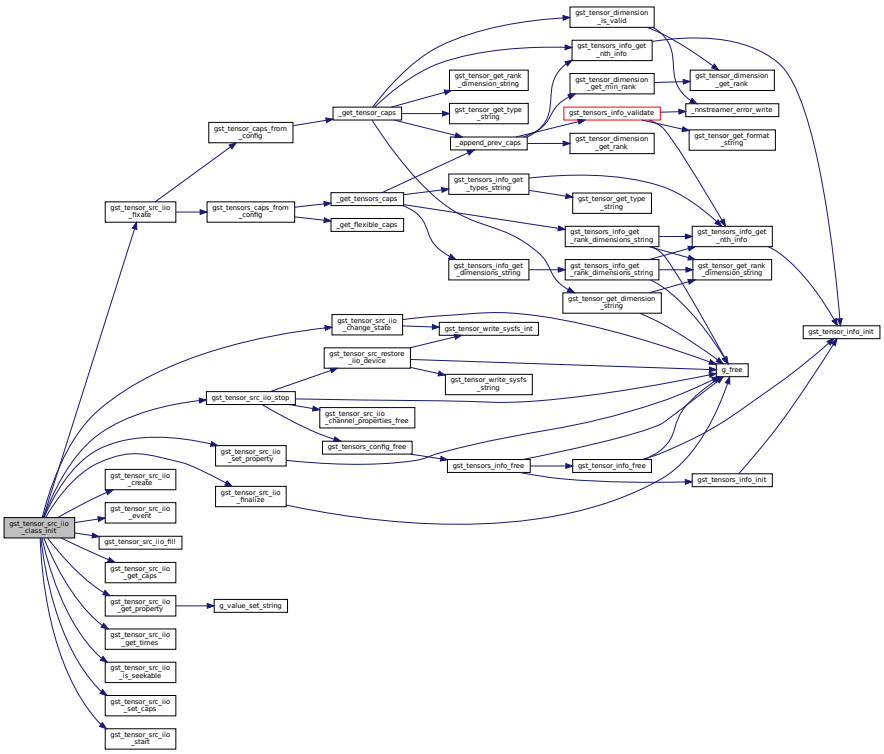
GObject methods

pad template

GstBaseSrcIIO methods

Definition at line 287 of file gsttensor_srciio.c.

Here is the call graph for this function:



9.64.4.11 gst_tensor_src_iio_create()

```
static GstFlowReturn gst_tensor_src_iio_create (
            GstBaseSrc * src,
            guint64 offset,
            guint size,
            GstBuffer ** buffer )  [static]
```

create a buffer with requested size and offset

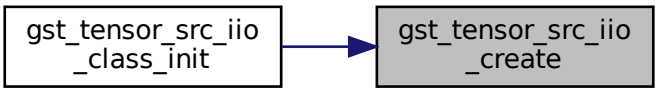
Note

offset, size ignored as the tensor src iio does not support pull mode

all the data, if unermged should be of the same size

Definition at line 2325 of file gsttensor_srciio.c.

Here is the caller graph for this function:



9.64.4.12 gst_tensor_src_iio_create_config()

```
static gboolean gst_tensor_src_iio_create_config (
            GstTensorSrcIIO * tensor_src_iio )  [static]
```

create the structure for the caps to update the src pad caps

Parameters

[in/out]	structure Caps structure which will filled

Returns

True if structure is created and filled, False for any error

create a bigger array, insert info in it and then merge tensors with same type+size

compile tensor info data

merge info about the tensors with same type

verify the merging of the array

tensors config data

buffer_capacity number of data samples are captured at once, packed together and sent downstream

Definition at line 1527 of file gsttensor_srciio.c.

9.64.4.13 gst_tensor_src_iio_device_properties_init()

```
static void gst_tensor_src_iio_device_properties_init (
            GstTensorSrcIIODeviceProperties * prop )  [static]
```

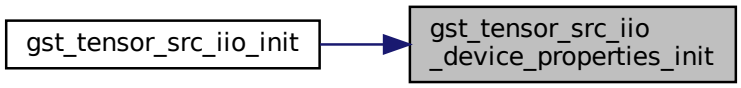
initialize GstTensorSrcIIODeviceProperties structure

Parameters

in	*data*	Device properties pointer to be initialized

Definition at line 427 of file gsttensor_srciio.c.

Here is the caller graph for this function:



9.64.4.14 gst_tensor_src_iio_event()

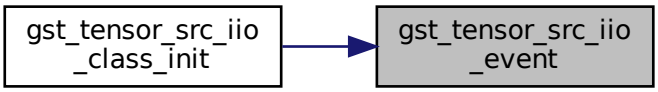
```
static gboolean gst_tensor_src_iio_event (
            GstBaseSrc * src,
            GstEvent * event )  [static]
```

handle events

No events to be handled yet

Definition at line 2132 of file gsttensor_srciio.c.

Here is the caller graph for this function:



9.64.4.15 gst_tensor_src_iio_fill()

```
static GstFlowReturn gst_tensor_src_iio_fill (
            GstBaseSrc * src,
            guint64 offset,
            guint size,
            GstBuffer * buffer )  [static]
```

fill the buffer with data

Note

ignore offset,size as there is pull mode

buffer timestamp is already handled by gstreamer with gst clock

Only supporting tensors made of 1 tensor for now

get writable buffer

memory to data from file

wait for the data to arrive

sleep for a device tick

using read for non-blocking access

parse the read data

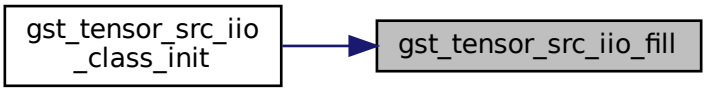
current assumption is that the all data is float and merged to form a 1 dimension data. 2nd dimension comes from buffer capacity.

for other/tensor, only 1 map exist as there is only 1 mem

for other/tensors, multiple maps exist as there are multiple mem

Definition at line 2462 of file gsttensor_srciio.c.

Here is the caller graph for this function:



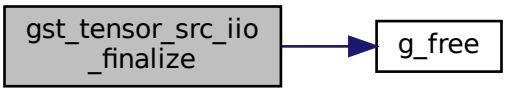
9.64.4.16 gst_tensor_src_iio_finalize()

```
static void gst_tensor_src_iio_finalize (
            GObject * object )  [static]
```

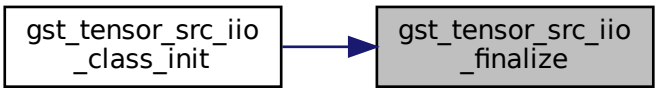
finalize the instance

Definition at line 1358 of file gsttensor_srciio.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.64.4.17 gst_tensor_src_iio_fixate()

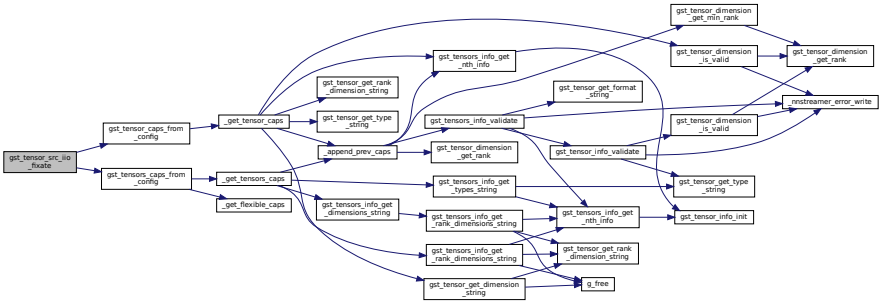
```
static GstCaps * gst_tensor_src_iio_fixate (
            GstBaseSrc * src,
            GstCaps * caps )  [static]
```

fixate the caps when needed during negotiation

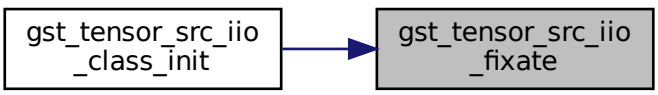
Caps are fixated based on the device source in _start().

Definition at line 2186 of file gsttensor_srciio.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.64.4.18 gst_tensor_src_iio_get_all_channel_info()

```
static gint gst_tensor_src_iio_get_all_channel_info (
            GstTensorSrcIIO * self,
            const gchar * dir_name )  [static]
```

get info about all the channels in the device

Parameters

	[in/out]	self Tensor src IIO object
in	*dir_name*	Directory name with all the scan elements for device

Returns

>=0 number of enabled channels -1 if any error when scanning channels

check for enable

not enabling and handling buffer timestamps for now

set the name and base_dir

find and set the current state

find and set the index

find and set the type information

if specific type info unavailable, use generic type info

find and setup offset info

find and setup scale info

sort the list with the order of the indices

Definition at line 860 of file gsttensor_srciio.c.

9.64.4.19 gst_tensor_src_iio_get_available_frequency()

```
static gint64 gst_tensor_src_iio_get_available_frequency (
            const gchar * base_dir,
            const guint64 frequency )  [static]
```

return sampling frequency given the frequency input from user

Parameters

in	*base_dir*	Device base directory (containing sampling freq file)
in	*frequency*	Frequency specified by user (else 0)

Returns

>0 if OK, represents sampling frequency to be set 0 if sampling frequency file does not exist, dont change anything -1 if any error occurs

get frequency list supported by the device

if the frequency is set 0, set the first available frequency else verify the frequency received from user is supported by the device

Definition at line 1036 of file gsttensor_srciio.c.

9.64.4.20 gst_tensor_src_iio_get_caps()

```
static GstCaps * gst_tensor_src_iio_get_caps (
            GstBaseSrc * src,
            GstCaps * filter )  [static]
```

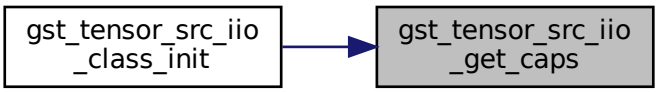
get caps of subclass

Note

basesrc _get_caps returns the caps from the pad_template however, we set the caps manually and needs to returned here

Definition at line 2160 of file gsttensor_srciio.c.

Here is the caller graph for this function:



9.64.4.21 gst_tensor_src_iio_get_float_from_file()

```
static gboolean gst_tensor_src_iio_get_float_from_file (
            const gchar * dirname,
            const gchar * name,
            const gchar * suffix,
            gfloat * value )  [static]
```

parse float value from the file

Parameters

in	*dirname*	Directory containing the file
in	*name*	Filename of the file
in	*suffix*	Suffix to be attached to the filename
	[in/out]	value Output value returned via value

Returns

FALSE on errors, else TRUE

Definition at line 677 of file gsttensor_srciio.c.

9.64.4.22 gst_tensor_src_iio_get_generic_name()

```
static gchar* gst_tensor_src_iio_get_generic_name (
            const gchar * channel_name )  [static]
```

get generic name for channel from the string

Parameters

in	*channel_name*	Name of the channel with its id embedded in it

Returns

Ptr to the generic name of the channel, caller should free the returned string

Definition at line 799 of file gsttensor_srciio.c.

9.64.4.23 gst_tensor_src_iio_get_id_by_name()

```
static gint gst_tensor_src_iio_get_id_by_name (
            const gchar * dir_name,
            const gchar * name,
            const gchar * prefix )  [static]
```

check if device/trigger with the given name exists

Parameters

in	*dir_name*	Directory containing all the devices
in	*name*	Name of the device to be found
in	*prefix*	Prefix to match with the filename of the device

Returns

>=0 if OK, represents device/trigger number -1 if returned with error

check for prefix and the next digit should be a number

Definition at line 576 of file gsttensor_srciio.c.

9.64.4.24 gst_tensor_src_iio_get_name_by_id()

```
static gchar* gst_tensor_src_iio_get_name_by_id (
            const gchar * dir_name,
            const gint id,
            const gchar * prefix )  [static]
```

check if device/trigger with the given id exists

Parameters

in	*dir_name*	Directory containing all the devices
in	*id*	ID of the device to be found
in	*prefix*	Prefix to match with the filename of the device

Returns

name on success (owned by caller), else NULL

Definition at line 640 of file gsttensor_srciio.c.

9.64.4.25 gst_tensor_src_iio_get_property()

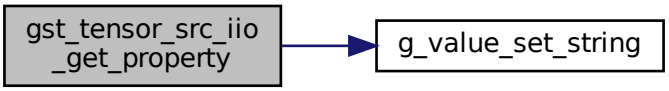
```
static void gst_tensor_src_iio_get_property (
            GObject * object,
            guint prop_id,
            GValue * value,
            GParamSpec * pspec )  [static]
```

get tensor_src_iio properties

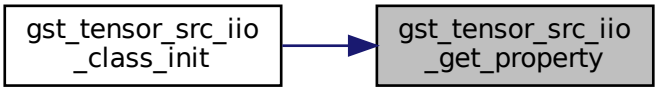
interface of frequency is kept long for outside but uint64 inside

Definition at line 1264 of file gsttensor_srciio.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.64.4.26 gst_tensor_src_iio_get_times()

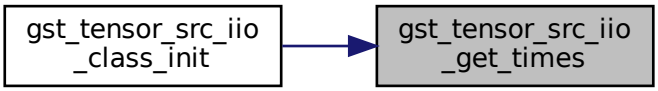
```
static void gst_tensor_src_iio_get_times (
             GstBaseSrc * basesrc,
             GstBuffer * buffer,
             GstClockTime * start,
             GstClockTime * end )  [static]
```

returns the time for the buffers

can't sync using DTS, use PTS

Definition at line 2298 of file gsttensor_srciio.c.

Here is the caller graph for this function:



9.64.4.27 gst_tensor_src_iio_init()

```
static void gst_tensor_src_iio_init (
             GstTensorSrcIIO * self )  [static]
```

initialize tensor_src_iio element.

init properties

format of the source since IIO device as a source is live and operates at a fixed frequency, GST_FORMAT_TIME is used

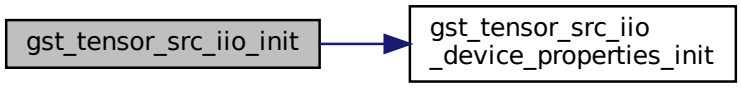
set the source to be live

set the timestamps on each buffer

set async is necessary to make state change async sync state changes does not need calling _start_complete() from _start()

Definition at line 439 of file gsttensor_srciio.c.

Here is the call graph for this function:



9.64.4.28 gst_tensor_src_iio_is_seekable()

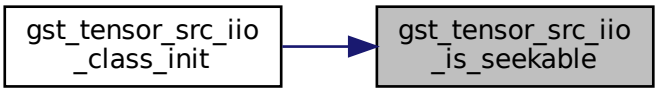
```
static gboolean gst_tensor_src_iio_is_seekable (
            GstBaseSrc * src )  [static]
```

check if source supports seeking

iio sensors are live source without any support for seeking

Definition at line 2287 of file gsttensor_srciio.c.

Here is the caller graph for this function:



9.64.4.29 gst_tensor_src_iio_process_scanned_data()

```
static gboolean gst_tensor_src_iio_process_scanned_data (
            GstTensorSrcIIOChannelProperties * prop,
            gchar * data,
            gfloat * buffer_map )  [static]
```

process the scanned data from IIO device

Parameters

in	*prop*	Properties of one of the enabled channels
in	*data*	Data read from the IIO device
	[in/out]	buffer_map Gst buffer map to write data to

Returns

FALSE if fail, else TRUE

assumes each data starting point is byte aligned right shift the extra storage bits

right shift the extra storage bits for big endian

mask out the extra storage bits for little endian

follow through

right shift the extra storage bits for big endian

mask out the extra storage bits for little endian

follow through

follow through

follow through

right shift the extra storage bits for big endian

mask out the extra storage bits for little endian

Definition at line 2376 of file gsttensor_srciio.c.

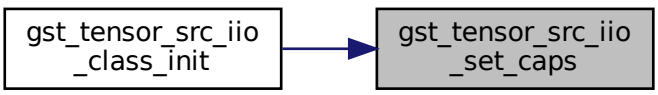
9.64.4.30 gst_tensor_src_iio_set_caps()

```
static gboolean gst_tensor_src_iio_set_caps (
            GstBaseSrc * src,
            GstCaps * caps )  [static]
```

set new caps

Definition at line 2142 of file gsttensor_srciio.c.

Here is the caller graph for this function:



9.64.4.31 gst_tensor_src_iio_set_channel_type()

```
static gboolean gst_tensor_src_iio_set_channel_type (
            GstTensorSrcIIOChannelProperties * prop,
            const gchar * contents )  [static]
```

get type info about the channel from the string

Parameters

	[in/out]	prop Channel properties where type info will be set
in	*contents*	Contains type unparsed information to be set

Returns

True if info was successfully set, false is info is not be parsed correctly @detail The format for the contents is expected to be of format

check endian

verify static parts of the contents

check sign

used bits

verify static parts of the contents

storage bits

verify static parts of the contents

Definition at line 718 of file gsttensor_srciio.c.

9.64.4.32 gst_tensor_src_iio_set_property()

```
static void gst_tensor_src_iio_set_property (
            GObject * object,
            guint prop_id,
            const GValue * value,
            GParamSpec * pspec )  [static]
```

set tensor_src_iio properties

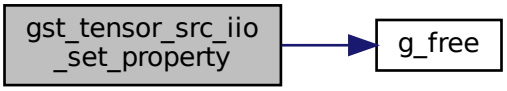
GObject method implementation No support for setting properties in PAUSED/PLAYING state as it needs to reset the device. To change the properties, user should stop the pipeline and set element state to READY/NULL and then change the properties

using direct as we only need to store keys and keys form a unique set

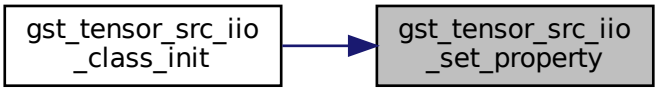
this means val is duplicated. just skip it, then.

Definition at line 1098 of file gsttensor_srciio.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.64.4.33 gst_tensor_src_iio_setup_device_buffer()

```
static gboolean gst_tensor_src_iio_setup_device_buffer (
            GstTensorSrcIIO * self )  [static]
```

setup device using name/id

Parameters

[in/out]	self Tensor src iio object

Returns

TRUE on success, FALSE on failure

once all these are set, set the buffer related thingies

open the buffer to read and ready the file descriptor

Definition at line 1913 of file gsttensor_srciio.c.

9.64.4.34 gst_tensor_src_iio_setup_device_properties()

```
static gboolean gst_tensor_src_iio_setup_device_properties (
            GstTensorSrcIIO * self )  [static]
```

setup device using name/id

Parameters

[in/out]	self Tensor src iio object

Returns

TRUE on success, FALSE on failure

Find the device

Definition at line 1623 of file gsttensor_srciio.c.

9.64.4.35 gst_tensor_src_iio_setup_sampling_frequency()

```
static gboolean gst_tensor_src_iio_setup_sampling_frequency (
            GstTensorSrcIIO * self )  [static]
```

setup device sampling frequency

Parameters

[in/out]	self Tensor src iio object

Returns

TRUE on success, FALSE on failure

check if sampling frequency file exists

reset the sampling frequency set by the user if any, as it cant be set

store the default frequency

verify the frequency given by the user if any from the list of available sampling frequencies

if sampling frequency file does not exist, no error

if sampling frequency file does not exist, sampling frequency is first value from the list of available sampling frequencies. So, we can ignore setting it

interface of frequency is kept long for outside but uint64 inside

Definition at line 1735 of file gsttensor_srciio.c.

9.64.4.36 gst_tensor_src_iio_setup_scan_channels()

```
static gboolean gst_tensor_src_iio_setup_scan_channels (
            GstTensorSrcIIO * self )  [static]
```

setup scan channels for the device

Parameters

[in/out]	self Tensor src iio object

Returns

TRUE on success, FALSE on failure

get all the channels that exist and then set enable on them

if enabling all channels failed, disable all channels

enable the custom channels and disable the rest

filter out disabled channels

set fixed caps for the src pad

create tensor_config

Definition at line 1822 of file gsttensor_srciio.c.

9.64.4.37 gst_tensor_src_iio_setup_trigger_properties()

```
static gboolean gst_tensor_src_iio_setup_trigger_properties (
            GstTensorSrcIIO * self )  [static]
```

setup trigger using name/id

Parameters

[in/out]	self Tensor src iio object

Returns

TRUE on success, FALSE on failure

register the trigger

verify if trigger is supported by our device

find if the provided trigger exists

get the default trigger, if any

set the trigger

Definition at line 1659 of file gsttensor_srciio.c.

9.64.4.38 gst_tensor_src_iio_start()

```
static gboolean gst_tensor_src_iio_start (
            GstBaseSrc * src )  [static]
```

start function, called when state changed null to ready. load the device and init the device resources

GstBaseSrc method implementation load and init resources

no support one shot mode for now

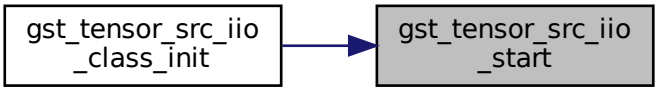
bytes every buffer will be fixed

complete the start of the base src

complete the start of the base src

Definition at line 1977 of file gsttensor_srciio.c.

Here is the caller graph for this function:



9.64.4.39 gst_tensor_src_iio_stop()

```
static gboolean gst_tensor_src_iio_stop (
            GstBaseSrc * src )  [static]
```

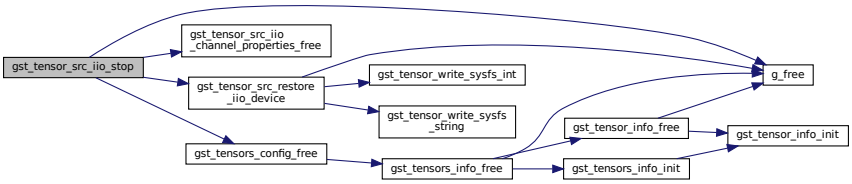
stop function, called when state changed ready to null.

free resources related to the device

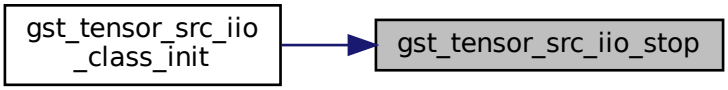
restore the iio device

Definition at line 2097 of file gsttensor_srciio.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.64.4.40 gst_tensor_src_merge_tensor_by_type()

```
static gint gst_tensor_src_merge_tensor_by_type (
            GstTensorInfo * info,
            guint size,
            guint dir )  [static]
```

merge multiple other/tensor

Note

they should have matching type and shape to form 1 other/tensors

extra dimension should be available for other/tensors

order of merge is stable

merging into 1 tensor only supported using innermost dimension.

Parameters

	[in/out]	info Tensor info to be merged
in	*size*	Info array size
in	*dir*	Innermost/outermost/innermost-outer (0/1/2) available dimension

Returns

>=0 number of valid entries in the info after merge -1 failed due to missing extra dimension/mismatch shape/type

base error control check

verify extra dimension (innermost to outermost)

verify that all the types and shapes match

return original if cant be merged and size within limits

If there are multiple available dimensions to merge along, we use dir to choose which the dimension to merge. If there is just 1 dimension, dir variable has no effect

No outer dimension available to merge

Now merge into 1 tensor using the selected dimension

Definition at line 491 of file gsttensor_srciio.c.

9.64.4.41 gst_tensor_src_restore_iio_device()

```
static void gst_tensor_src_restore_iio_device (
             GstTensorSrcIIO * self )  [static]
```

restore the iio device to its original device.

reset enabled channels

reset sampling_frequency

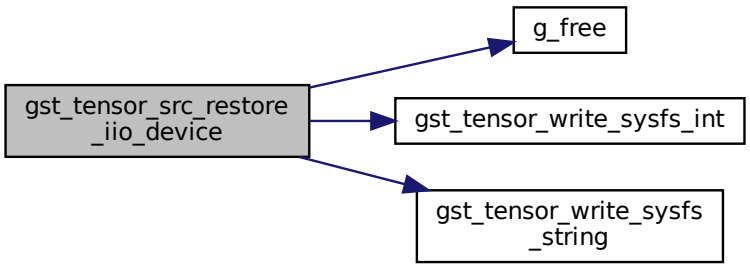
converting to long as setting interface to device

reset buffer_capacity

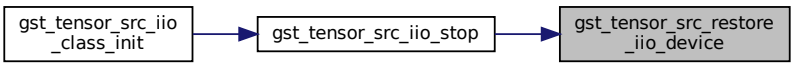
reset current trigger

Definition at line 2049 of file gsttensor_srciio.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.64.4.42 gst_tensor_write_sysfs_int()

```
static gboolean gst_tensor_write_sysfs_int (
            GstTensorSrcIIO * self,
            const gchar * file,
            const gchar * base_dir,
            const gint contents )  [static]
```

write the int in to the file

Parameters

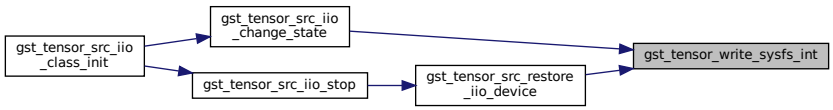
in	*self*	Tensor src IIO object
in	*file*	Destination file for the data
in	*base_dir*	Directory containing the file
in	*contents*	Data to be written to the file

Returns

True if write was successful, false on failure

Definition at line 1449 of file gsttensor_srciio.c.

Here is the caller graph for this function:



9.64.4.43 gst_tensor_write_sysfs_string()

```
static gboolean gst_tensor_write_sysfs_string (
            GstTensorSrcIIO * self,
            const gchar * file,
            const gchar * base_dir,
            const gchar * contents )  [static]
```

write the string in to the file

Parameters

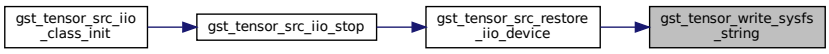
in	*self*	Tensor src IIO object
in	*file*	Destination file for the data
in	*base_dir*	Directory containing the file
in	*contents*	Data to be written to the file

Returns

True if write was successful, false on failure

Definition at line 1384 of file gsttensor_srciio.c.

Here is the caller graph for this function:



9.64.4.44 if()

```
value_unsigned& if (
            prop-> is_signed )
```

Definition at line 117 of file gsttensor_srciio.c.

9.64.4.45 PROCESS_SCANNED_DATA() [1/4]

```
PROCESS_SCANNED_DATA (
             guint16 ,
             gint16  )
```

9.64.4.46 PROCESS_SCANNED_DATA() [2/4]

```
PROCESS_SCANNED_DATA (
             guint32 ,
             gint32  )
```

9.64.4.47 PROCESS_SCANNED_DATA() [3/4]

```
PROCESS_SCANNED_DATA (
             guint64 ,
             gint64  )
```

9.64.4.48 PROCESS_SCANNED_DATA() [4/4]

```
PROCESS_SCANNED_DATA (
             guint8 ,
             gint8  )
```

Define data processing functions for various types

9.64.5 Variable Documentation

9.64.5.1 else

```
else
```

Initial value:

```
{
    value_float = ((gfloat) value_unsigned + prop->offset) * prop->scale
```

Definition at line 123 of file gsttensor_srciio.c.

9.64.5.2 prop

```
GstTensorSrcIIOChannelProperties* prop
```

DTYPE_UNSIGNED (.

Definition at line 110 of file gsttensor_srciio.c.

9.64.5.3 value_float

```
return value_float
```

Definition at line 126 of file gsttensor_srciio.c.

9.64.5.4 value_unsigned

```
value_unsigned
```

Initial value:

```
{
  gfloat value_float
```

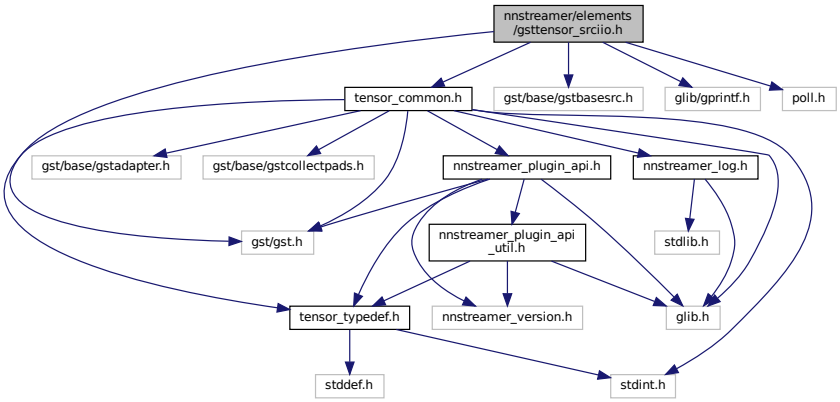
Definition at line 110 of file gsttensor_srciio.c.

9.65 nnstreamer/elements/gsttensor_srciio.h File Reference

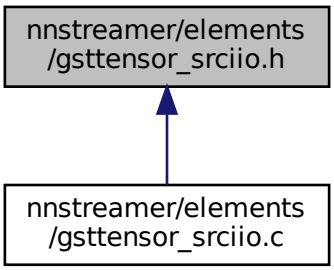
GStreamer plugin to support linux IIO as tensor(s)

```
#include <gst/gst.h>
#include <gst/base/gstbasesrc.h>
#include <glib/gprintf.h>
#include <tensor_common.h>
#include <poll.h>
```

Include dependency graph for gsttensor_srciio.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct _GstTensorSrcIIODeviceProperties

  GstTensorSrcIIO devices's properties (internal data structure)
- struct _GstTensorSrcIIOChannelProperties

  GstTensorSrcIIO channel's properties (internal data structure)
- struct _GstTensorSrcIIO

  GstTensorSrcIIO data structure.
- struct _GstTensorSrcIIOClass

  GstTensorSrcIIOClass data structure.

Macros

- #define GST_TYPE_TENSOR_SRC_IIO (gst_tensor_src_iio_get_type())
- #define GST_TENSOR_SRC_IIO(obj) (G_TYPE_CHECK_INSTANCE_CAST((obj),GST_TYPE_TENSOR_SRC_IIO,GstTenso
- #define GST_TENSOR_SRC_IIO_CLASS(klass) (G_TYPE_CHECK_CLASS_CAST((klass),GST_TYPE_TENSOR_SRC_IIO,
- #define GST_IS_TENSOR_SRC_IIO(obj) (G_TYPE_CHECK_INSTANCE_TYPE((obj),GST_TYPE_TENSOR_SRC_IIO))
- #define GST_IS_TENSOR_SRC_IIO_CLASS(klass) (G_TYPE_CHECK_CLASS_TYPE((klass),GST_TYPE_TENSOR_SRC_I
- #define GST_TENSOR_SRC_IIO_CAST(obj) ((GstTensorSrcIIO *)(obj))

Typedefs

- typedef struct _GstTensorSrcIIO GstTensorSrcIIO
- typedef struct _GstTensorSrcIIOClass GstTensorSrcIIOClass
- typedef struct _GstTensorSrcIIODeviceProperties GstTensorSrcIIODeviceProperties

  GstTensorSrcIIO devices's properties (internal data structure)
- typedef struct _GstTensorSrcIIOChannelProperties GstTensorSrcIIOChannelProperties

  GstTensorSrcIIO channel's properties (internal data structure)

Enumerations

- enum channels_enabled_options { CHANNELS_ENABLED_ALL, CHANNELS_ENABLED_AUTO, CHANNELS_ENABLED_CUSTOM }

  iio device channel enabled mode

Functions

- GType gst_tensor_src_iio_get_type (void)

  Function to get type of tensor_src_iio.

9.65.1 Detailed Description

GStreamer plugin to support linux IIO as tensor(s)

GStreamer Tensor_Src_IIO Copyright (C) 2005 Thomas Vander Stichele thomas@apestaart.org Copyright (C) 2005 Ronald S. Bultje rbultje@ronald.bitfreak.net Copyright (C) 2019 Parichay Kapoor pk.kapoor@samsung.com

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Library General Public License as published by the Free Software Foundation; version 2.1 of the License.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Library General Public License for more details.

Date

26 Feb 2019

See also

https://github.com/nnstreamer/nnstreamer

Author

Parichay Kapoor pk.kapoor@samsung.com

Bug No known bugs except for NYI items

9.65.2 Macro Definition Documentation

9.65.2.1 GST_IS_TENSOR_SRC_IIO

```
#define GST_IS_TENSOR_SRC_IIO(
              obj ) (G_TYPE_CHECK_INSTANCE_TYPE((obj),GST_TYPE_TENSOR_SRC_IIO))
```

Definition at line 43 of file gsttensor_srciio.h.

9.65.2.2 GST_IS_TENSOR_SRC_IIO_CLASS

```
#define GST_IS_TENSOR_SRC_IIO_CLASS(
              klass ) (G_TYPE_CHECK_CLASS_TYPE((klass),GST_TYPE_TENSOR_SRC_IIO))
```

Definition at line 45 of file gsttensor_srciio.h.

9.65.2.3 GST_TENSOR_SRC_IIO

```
#define GST_TENSOR_SRC_IIO(
              obj ) (G_TYPE_CHECK_INSTANCE_CAST((obj),GST_TYPE_TENSOR_SRC_IIO,GstTensorSrcIIO))
```

Definition at line 39 of file gsttensor_srciio.h.

9.65.2.4 GST_TENSOR_SRC_IIO_CAST

```
#define GST_TENSOR_SRC_IIO_CAST(
              obj ) ((GstTensorSrcIIO *)(obj))
```

Definition at line 47 of file gsttensor_srciio.h.

9.65.2.5 GST_TENSOR_SRC_IIO_CLASS

```
#define GST_TENSOR_SRC_IIO_CLASS(
              klass ) (G_TYPE_CHECK_CLASS_CAST((klass),GST_TYPE_TENSOR_SRC_IIO,GstTensorSrcIIOClass))
```

Definition at line 41 of file gsttensor_srciio.h.

9.65.2.6 GST_TYPE_TENSOR_SRC_IIO

```
#define GST_TYPE_TENSOR_SRC_IIO (gst_tensor_src_iio_get_type())
```

Definition at line 37 of file gsttensor_srciio.h.

9.65.3 Typedef Documentation

9.65.3.1 GstTensorSrcIIO

```
typedef struct _GstTensorSrcIIO GstTensorSrcIIO
```

Definition at line 48 of file gsttensor_srciio.h.

9.65.3.2 GstTensorSrcIIOChannelProperties

```
typedef struct _GstTensorSrcIIOChannelProperties GstTensorSrcIIOChannelProperties
```

GstTensorSrcIIO channel's properties (internal data structure)

9.65.3.3 GstTensorSrcIIOClass

```
typedef struct _GstTensorSrcIIOClass GstTensorSrcIIOClass
```

Definition at line 49 of file gsttensor_srciio.h.

9.65.3.4 GstTensorSrcIIODeviceProperties

```
typedef struct _GstTensorSrcIIODeviceProperties GstTensorSrcIIODeviceProperties
```

GstTensorSrcIIO devices's properties (internal data structure)

This data structure is used for both device/triggers, as triggers are also iio devices

9.65.4 Enumeration Type Documentation

9.65.4.1 channels_enabled_options

```
enum channels_enabled_options
```

iio device channel enabled mode

Enumerator

CHANNELS_ENABLED_ALL	
CHANNELS_ENABLED_AUTO	
CHANNELS_ENABLED_CUSTOM	

Definition at line 54 of file gsttensor_srciio.h.

9.65.5 Function Documentation

9.65.5.1 gst_tensor_src_iio_get_type()

```
GType gst_tensor_src_iio_get_type (
            void  )
```

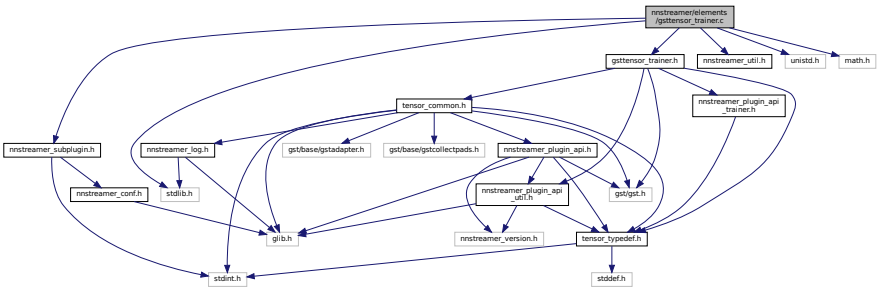
Function to get type of tensor_src_iio.

9.66 nnstreamer/elements/gsttensor_trainer.c File Reference

GStreamer plugin to train tensor data using NN Frameworks.

```
#include <stdlib.h>
#include <nnstreamer_subplugin.h>
#include <nnstreamer_util.h>
#include "gsttensor_trainer.h"
#include <unistd.h>
#include <math.h>
```

Include dependency graph for gsttensor_trainer.c:



Macros

- #define SINK_CAPS_STRING GST_TENSORS_CAP_MAKE ("{ static, flexible }")

  Default caps string for sink.
- #define SRC_CAPS_STRING GST_TENSORS_CAP_MAKE ("{ static}")

  Default caps string for src.
- #define GST_CAT_DEFAULT gst_tensor_trainer_debug
- #define gst_tensor_trainer_parent_class parent_class
- #define MODEL_STATS_SIZE 4
- #define DEFAULT_PROP_INPUT_LIST 1

  Default framework property value.
- #define DEFAULT_PROP_LABEL_LIST 1
- #define DEFAULT_PROP_TRAIN_SAMPLES 0
- #define DEFAULT_PROP_VALID_SAMPLES 0
- #define DEFAULT_PROP_EPOCHS 1
- #define DEFAULT_STR_PROP_VALUE ""

  Default string property value.

Enumerations

- enum { TRAINING_LOSS, TRAINING_ACCURACY, VALIDATION_LOSS, VALIDATION_ACCURACY }

  Statistical from the model being trained An enum value indicates the value stored at the index of the output tensor.
- enum {
  PROP_0, PROP_FRAMEWORK, PROP_MODEL_CONFIG, PROP_MODEL_SAVE_PATH,
  PROP_MODEL_LOAD_PATH, PROP_NUM_INPUTS, PROP_NUM_LABELS, PROP_NUM_TRAINING_SAMPLES,
  PROP_NUM_VALIDATION_SAMPLES, PROP_EPOCHS }

  tensor_trainer properties

Functions

- GST_DEBUG_CATEGORY_STATIC (gst_tensor_trainer_debug)
- G_DEFINE_TYPE (GstTensorTrainer, gst_tensor_trainer, GST_TYPE_ELEMENT)
- static void gst_tensor_trainer_set_property (GObject *object, guint prop_id, const GValue *value, GParam↩ Spec *pspec)

  Setter for tensor_trainsink properties.
- static void gst_tensor_trainer_get_property (GObject *object, guint prop_id, GValue *value, GParamSpec *pspec)

  Getter tensor_trainsink properties.
- static void gst_tensor_trainer_finalize (GObject *object)

  Function to finalize instance.
- static gboolean gst_tensor_trainer_sink_event (GstPad *sinkpad, GstObject *parent, GstEvent *event)

  Event handler for sink pad of tensor_trainer.
- static gboolean gst_tensor_trainer_sink_query (GstPad *sinkpad, GstObject *parent, GstQuery *query)

  This function handles sink pad query.
- static gboolean gst_tensor_trainer_src_query (GstPad *srcpad, GstObject *parent, GstQuery *query)

  This function handles src pad query.
- static GstFlowReturn gst_tensor_trainer_chain (GstPad *sinkpad, GstObject *parent, GstBuffer *inbuf)

  Chain function, this function does the actual processing.
- static GstCaps * gst_tensor_trainer_query_caps (GstTensorTrainer *trainer, GstPad *pad, GstCaps *filter)

  Get pad caps for caps negotiation.
- static GstStateChangeReturn gst_tensor_trainer_change_state (GstElement *element, GstStateChange transition)

  Change state of tensor_trainsink.
- static void gst_tensor_trainer_set_prop_framework (GstTensorTrainer *trainer, const GValue *value)

  Handle "PROP_FRAMEWORK" for set-property.
- static void gst_tensor_trainer_set_prop_model_config_file_path (GstTensorTrainer *trainer, const GValue *value)

  Handle "PROP_MODEL_CONFIG" for set-property.
- static void gst_tensor_trainer_set_model_save_path (GstTensorTrainer *trainer, const GValue *value)

  Handle "PROP_MODEL_SAVE_PATH" for set-property.
- static void gst_tensor_trainer_set_model_load_path (GstTensorTrainer *trainer, const GValue *value)

  Handle "PROP_MODEL_LOAD_PATH" for set-property.
- static gboolean gst_tensor_trainer_find_framework (GstTensorTrainer *trainer, const char *name)

  Find Trainer sub-plugin with the name.
- static gboolean gst_tensor_trainer_create_framework (GstTensorTrainer *trainer)

  Create NN framework.
- static gsize gst_tensor_trainer_get_tensor_size (GstTensorTrainer *trainer, guint index, gboolean is_input)

  Calculate tensor buffer size.
- static gboolean gst_tensor_trainer_create_model (GstTensorTrainer *trainer)

Create model.

- static void gst_tensor_trainer_create_event_notifier (GstTensorTrainer *trainer)

  Create a event notifier.
- static void gst_tensor_trainer_start_model_training (GstTensorTrainer *trainer)

  Start model training.
- static void gst_tensor_trainer_stop_model_training (GstTensorTrainer *trainer)

  Stop model training.
- static void gst_tensor_trainer_set_output_meta (GstTensorTrainer *trainer)

  initialize the output tensor dimension
- static void gst_tensor_trainer_class_init (GstTensorTrainerClass *klass)

  initialize the tensor_trainer's class
- static void gst_tensor_trainer_init (GstTensorTrainer *trainer)

  Initialize tensor_trainer.
- static gboolean gst_tensor_trainer_check_invalid_param (GstTensorTrainer *trainer)

  Check invalid param.
- static gpointer gst_tensor_trainer_dummy_data_generation_func (GstTensorTrainer *trainer)

  Dummy data generation thread.
- static void gst_tensor_trainer_wait_for_epoch_completion (GstTensorTrainer *trainer)

  Wait for epoch eompletion.
- static gboolean gst_tensor_trainer_epochs_is_complete (GstTensorTrainer *trainer)

  Check if current epochs is complete, tensor_trainer wait for one of epochs to complete before getting the results from the subplugin.
- static gboolean gst_tensor_trainer_check_buffer_drop_conditions (GstTensorTrainer *trainer)

  Check buffer drop conditions. If condition is met, drop the buffer.
- static gboolean gst_tensor_trainer_check_chain_conditions (GstTensorTrainer *trainer, guint num_tensors)

  Check chain conditions. If all conditions are met, proceed to next step.
- static gsize gst_tensor_trainer_convert_meta (GstTensorTrainer *trainer, GstTensorMetaInfo *meta, GstTensorInfo *info, void *data)

  Convert tensor meta and get the size of tensor header.
- static gboolean gst_tensor_trainer_push_input (GstTensorTrainer *trainer, GstBuffer *inbuf, gboolean in_↩ flexible)

  Create input tensors from the buffer and push it into trainer fw.
- static gboolean gst_tensor_trainer_get_model_stats (GstTensorTrainer *trainer, double *model_stats)

  Get the model statistics from the sub-plugin.
- static GstBuffer * gst_tensor_trainer_create_output (GstTensorTrainer *trainer)

  Create output tensors.
- static void gst_tensor_trainer_wait_for_training_completion (GstTensorTrainer *trainer)

  Wait for training completion.
- int nnstreamer_trainer_probe (GstTensorTrainerFramework *ttsp)

  Trainer's sub-plugin should call this function to register itself.
- int nnstreamer_trainer_exit (GstTensorTrainerFramework *ttsp)

  Trainer's sub-plugin may call this to unregister itself.
- void nnstreamer_trainer_notify_event (GstTensorTrainerEventNotifier *notifier, GstTensorTrainerEventType type, void *data)

  Trainer's sub-plugin may call this to send event.

Variables

- static GstStaticPadTemplate sink_template

  The capabilities of the sink pad.
- static GstStaticPadTemplate src_template

  The capabilities of the src pad.

9.66.1 Detailed Description

GStreamer plugin to train tensor data using NN Frameworks.

Date

20 October 2022

See also

https://github.com/nnstreamer/nnstreamer

Author

Hyunil Park hyunil46.park@samsung.com

Bug No known bugs except for NYI items

//! Example launch line |[gst-launch-1.0 datareposrc location=mnist_trainingSet.dat json=mnist.json start-sample-index=3 stop-sample-index=202 epochs=5 ! \ tensor_trainer framework=nntrainer model-config=mnist.ini model-save-path=model.bin \ num-inputs=1 num-labels=1 num-training-samples=100 num-validation-samples=100 epochs=5 ! \ tensor_sink]|

Total number of data to be received is 1000((num-training-samples + num-validation-samples) $*$ epochs)

output tensors : dimensions=1:1:4, types=float64. values are training loss, training accuracy, validation loss and validation accuracy. -INFINITY value is stored if the value fetched from the sub-plugin is not greater than 0.

9.66.2 Macro Definition Documentation

9.66.2.1 DEFAULT_PROP_EPOCHS

```
#define DEFAULT_PROP_EPOCHS 1
```

Definition at line 88 of file gsttensor_trainer.c.

9.66.2.2 DEFAULT_PROP_INPUT_LIST

```
#define DEFAULT_PROP_INPUT_LIST 1
```

Default framework property value.

Definition at line 84 of file gsttensor_trainer.c.

9.66.2.3 DEFAULT_PROP_LABEL_LIST

```
#define DEFAULT_PROP_LABEL_LIST 1
```

Definition at line 85 of file gsttensor_trainer.c.

9.66.2.4 DEFAULT_PROP_TRAIN_SAMPLES

```
#define DEFAULT_PROP_TRAIN_SAMPLES 0
```

Definition at line 86 of file gsttensor_trainer.c.

9.66.2.5 DEFAULT_PROP_VALID_SAMPLES

```
#define DEFAULT_PROP_VALID_SAMPLES 0
```

Definition at line 87 of file gsttensor_trainer.c.

9.66.2.6 DEFAULT_STR_PROP_VALUE

```
#define DEFAULT_STR_PROP_VALUE ""
```

Default string property value.

Definition at line 92 of file gsttensor_trainer.c.

9.66.2.7 GST_CAT_DEFAULT

```
#define GST_CAT_DEFAULT gst_tensor_trainer_debug
```

Definition at line 64 of file gsttensor_trainer.c.

9.66.2.8 gst_tensor_trainer_parent_class

```
#define gst_tensor_trainer_parent_class parent_class
```

Definition at line 65 of file gsttensor_trainer.c.

9.66.2.9 MODEL_STATS_SIZE

```
#define MODEL_STATS_SIZE 4
```

Definition at line 79 of file gsttensor_trainer.c.

9.66.2.10 SINK_CAPS_STRING

```
#define SINK_CAPS_STRING GST_TENSORS_CAP_MAKE ("{ static, flexible }")
```

Default caps string for sink.

Definition at line 40 of file gsttensor_trainer.c.

9.66.2.11 SRC_CAPS_STRING

```
#define SRC_CAPS_STRING GST_TENSORS_CAP_MAKE ("{ static}")
```

Default caps string for src.

Definition at line 45 of file gsttensor_trainer.c.

9.66.3 Enumeration Type Documentation

9.66.3.1 anonymous enum

```
anonymous enum
```

Statistical from the model being trained An enum value indicates the value stored at the index of the output tensor.

Enumerator

TRAINING_LOSS	
TRAINING_ACCURACY	
VALIDATION_LOSS	
VALIDATION_ACCURACY	

Definition at line 72 of file gsttensor_trainer.c.

9.66.3.2 anonymous enum

```
anonymous enum
```

tensor_trainer properties

Enumerator

PROP_0	
PROP_FRAMEWORK	
PROP_MODEL_CONFIG	
PROP_MODEL_SAVE_PATH	
PROP_MODEL_LOAD_PATH	
PROP_NUM_INPUTS	
PROP_NUM_LABELS	
PROP_NUM_TRAINING_SAMPLES	
PROP_NUM_VALIDATION_SAMPLES	
PROP_EPOCHS	

Definition at line 97 of file gsttensor_trainer.c.

9.66.4 Function Documentation

9.66.4.1 G_DEFINE_TYPE()

```
G_DEFINE_TYPE (
             GstTensorTrainer ,
             gst_tensor_trainer ,
             GST_TYPE_ELEMENT  )
```

9.66.4.2 GST_DEBUG_CATEGORY_STATIC()

```
GST_DEBUG_CATEGORY_STATIC (
             gst_tensor_trainer_debug  )
```

9.66.4.3 gst_tensor_trainer_chain()

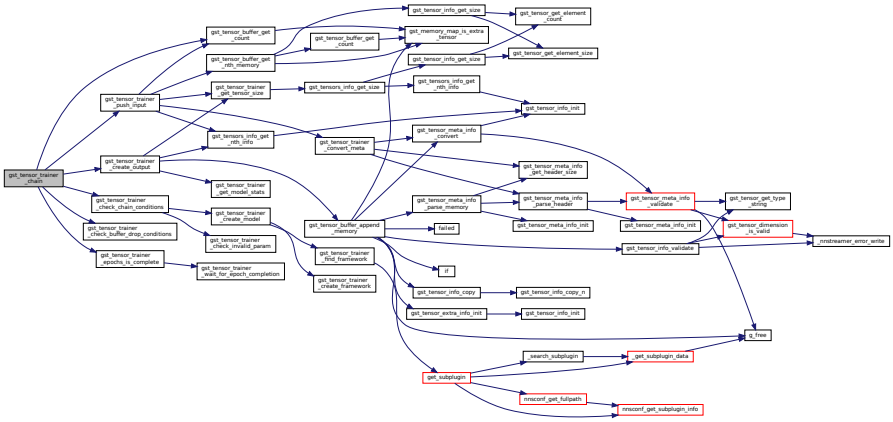
```
static GstFlowReturn gst_tensor_trainer_chain (
            GstPad * sinkpad,
            GstObject * parent,
            GstBuffer * inbuf )  [static]
```

Chain function, this function does the actual processing.

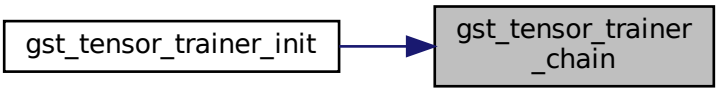
Update result if one of epochs is complete, push one outbuf is necessary to change pipeline state. Scheduling with subplugin does not work.

Definition at line 868 of file gsttensor_trainer.c.

Here is the call graph for this function:



Here is the caller graph for this function:



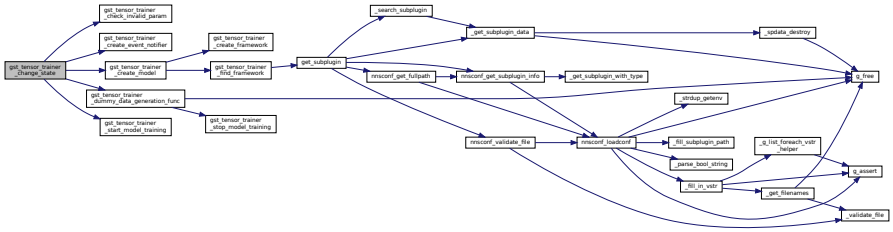
9.66.4.4 gst_tensor_trainer_change_state()

```
static GstStateChangeReturn gst_tensor_trainer_change_state (
            GstElement * element,
            GstStateChange transition )  [static]
```

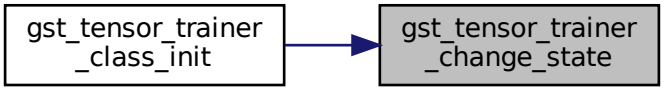
Change state of tensor_trainsink.

Definition at line 513 of file gsttensor_trainer.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.66.4.5 gst_tensor_trainer_check_buffer_drop_conditions()

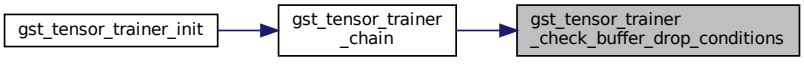
```
static gboolean gst_tensor_trainer_check_buffer_drop_conditions (
            GstTensorTrainer * trainer )  [static]
```

Check buffer drop conditions. If condition is met, drop the buffer.

app need to send gst_element_send_event(tensor_trainer, gst_event_new_eos()) after training_complete or set eos to datareposrc

Definition at line 629 of file gsttensor_trainer.c.

Here is the caller graph for this function:



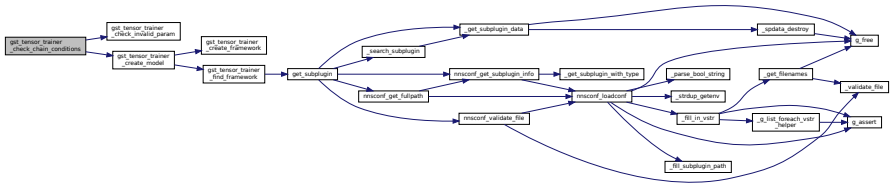
9.66.4.6 gst_tensor_trainer_check_chain_conditions()

```
static gboolean gst_tensor_trainer_check_chain_conditions (
            GstTensorTrainer * trainer,
            guint num_tensors )  [static]
```

Check chain conditions. If all conditions are met, proceed to next step.

Definition at line 645 of file gsttensor_trainer.c.

Here is the call graph for this function:



Here is the caller graph for this function:

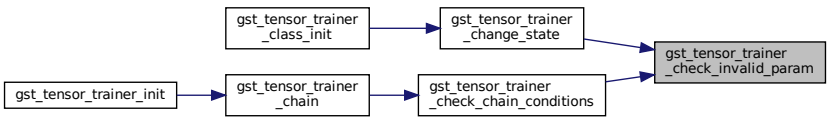
9.66.4.7 gst_tensor_trainer_check_invalid_param()

```
static gboolean gst_tensor_trainer_check_invalid_param (
            GstTensorTrainer * trainer )  [static]
```

Check invalid param.

Definition at line 440 of file gsttensor_trainer.c.

Here is the caller graph for this function:



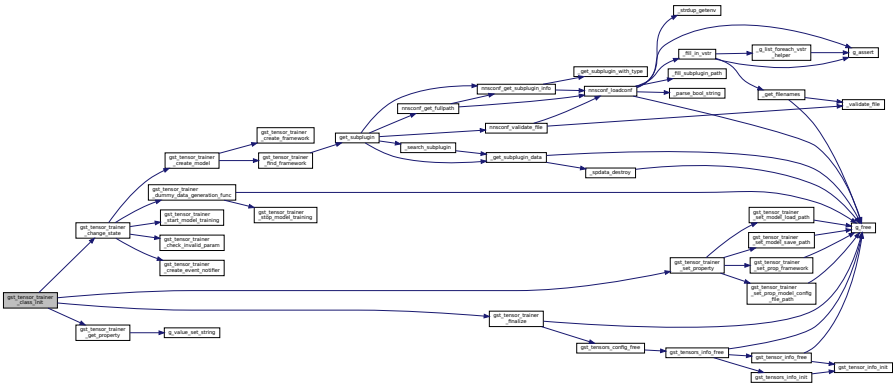
9.66.4.8 gst_tensor_trainer_class_init()

```
static void gst_tensor_trainer_class_init (
            GstTensorTrainerClass * klass )  [static]
```

initialize the tensor_trainer's class

Definition at line 155 of file gsttensor_trainer.c.

Here is the call graph for this function:



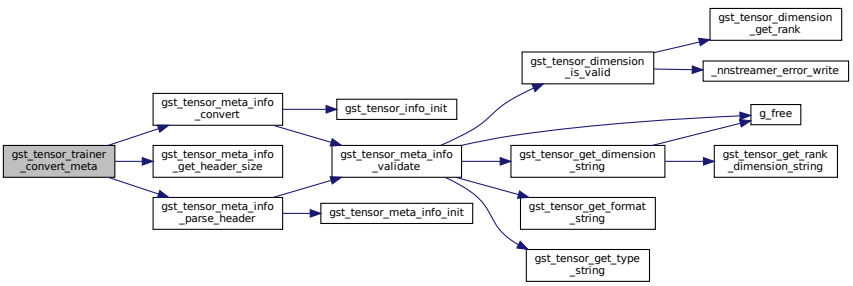
9.66.4.9 gst_tensor_trainer_convert_meta()

```
static gsize gst_tensor_trainer_convert_meta (
            GstTensorTrainer * trainer,
            GstTensorMetaInfo * meta,
            GstTensorInfo * info,
            void * data )  [static]
```

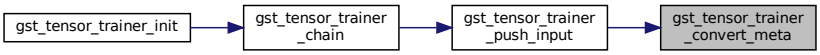
Convert tensor meta and get the size of tensor header.

Definition at line 665 of file gsttensor_trainer.c.

Here is the call graph for this function:



Here is the caller graph for this function:



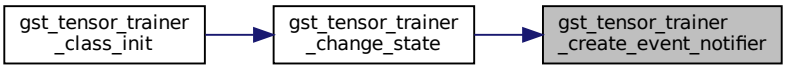
9.66.4.10 gst_tensor_trainer_create_event_notifier()

```
static void gst_tensor_trainer_create_event_notifier (
            GstTensorTrainer * trainer )  [static]
```

Create a event notifier.

Definition at line 1296 of file gsttensor_trainer.c.

Here is the caller graph for this function:



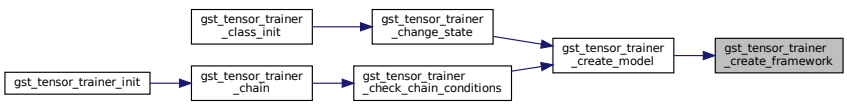
9.66.4.11 gst_tensor_trainer_create_framework()

```
static gboolean gst_tensor_trainer_create_framework (
            GstTensorTrainer * trainer )  [static]
```

Create NN framework.

Definition at line 1221 of file gsttensor_trainer.c.

Here is the caller graph for this function:



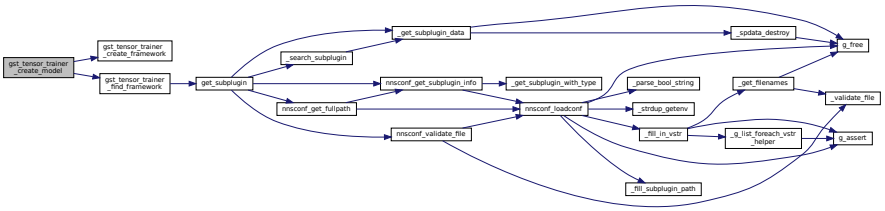
9.66.4.12 gst_tensor_trainer_create_model()

```
static gboolean gst_tensor_trainer_create_model (
            GstTensorTrainer * trainer )  [static]
```

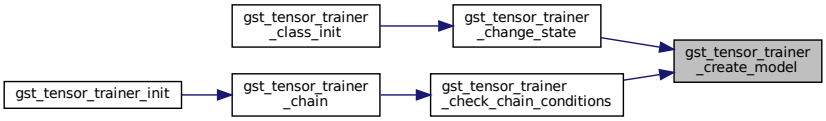
Create model.

Definition at line 1273 of file gsttensor_trainer.c.

Here is the call graph for this function:



Here is the caller graph for this function:



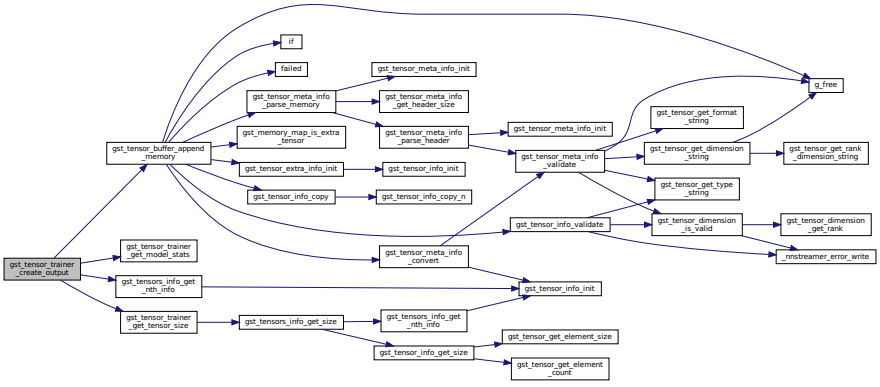
9.66.4.13 gst_tensor_trainer_create_output()

```
static GstBuffer* gst_tensor_trainer_create_output (
            GstTensorTrainer * trainer )  [static]
```

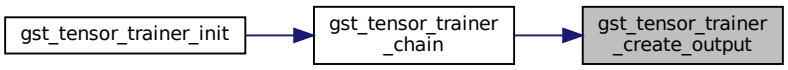
Create output tensors.

Definition at line 805 of file gsttensor_trainer.c.

Here is the call graph for this function:



Here is the caller graph for this function:



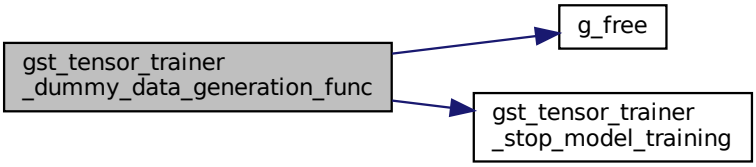
9.66.4.14 gst_tensor_trainer_dummy_data_generation_func()

```
static gpointer gst_tensor_trainer_dummy_data_generation_func (
            GstTensorTrainer * trainer )  [static]
```

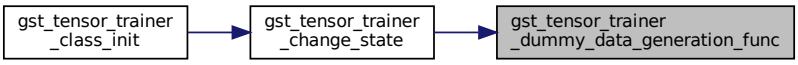
Dummy data generation thread.

Definition at line 471 of file gsttensor_trainer.c.

Here is the call graph for this function:



Here is the caller graph for this function:



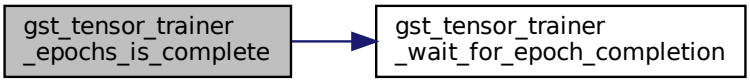
9.66.4.15 gst_tensor_trainer_epochs_is_complete()

```
static gboolean gst_tensor_trainer_epochs_is_complete (
            GstTensorTrainer * trainer )  [static]
```

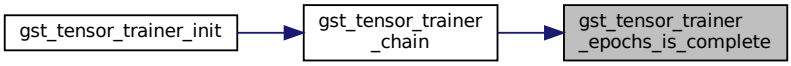
Check if current epochs is complete, tensor_trainer wait for one of epochs to complete before getting the results from the subplugin.

Definition at line 609 of file gsttensor_trainer.c.

Here is the call graph for this function:



Here is the caller graph for this function:



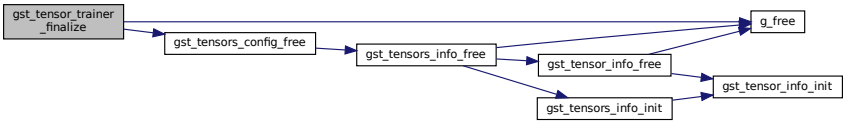
9.66.4.16 gst_tensor_trainer_finalize()

```
static void gst_tensor_trainer_finalize (
            GObject * object )  [static]
```

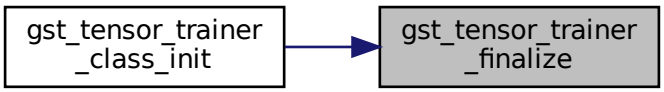
Function to finalize instance.

Definition at line 316 of file gsttensor_trainer.c.

Here is the call graph for this function:



Here is the caller graph for this function:



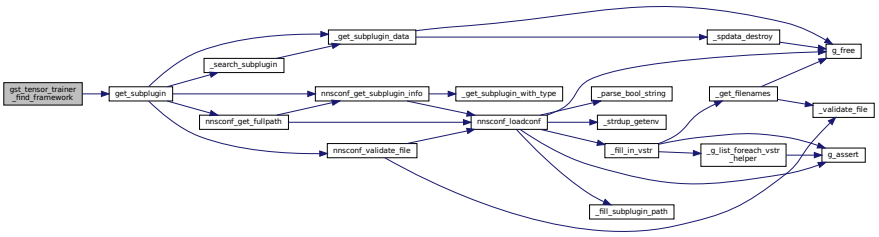
9.66.4.17 gst_tensor_trainer_find_framework()

```
static gboolean gst_tensor_trainer_find_framework (
            GstTensorTrainer * trainer,
            const char * name )  [static]
```

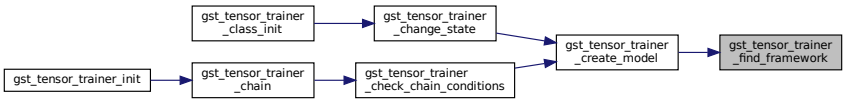
Find Trainer sub-plugin with the name.

Definition at line 1196 of file gsttensor_trainer.c.

Here is the call graph for this function:



Here is the caller graph for this function:



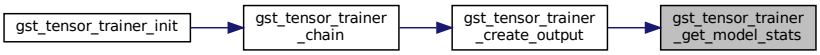
9.66.4.18 gst_tensor_trainer_get_model_stats()

```
static gboolean gst_tensor_trainer_get_model_stats (
            GstTensorTrainer * trainer,
            double * model_stats )  [static]
```

Get the model statistics from the sub-plugin.

Definition at line 769 of file gsttensor_trainer.c.

Here is the caller graph for this function:



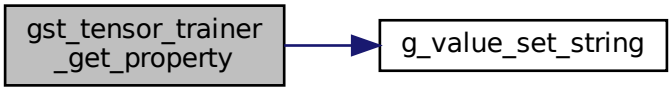
9.66.4.19 gst_tensor_trainer_get_property()

```
static void gst_tensor_trainer_get_property (
            GObject * object,
            guint prop_id,
            GValue * value,
            GParamSpec * pspec )  [static]
```

Getter tensor_trainsink properties.

Definition at line 395 of file gsttensor_trainer.c.

Here is the call graph for this function:



Here is the caller graph for this function:

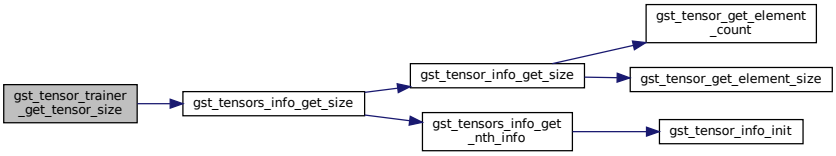
9.66.4.20 gst_tensor_trainer_get_tensor_size()

```
gsize gst_tensor_trainer_get_tensor_size (
            GstTensorTrainer * trainer,
            guint index,
            gboolean is_input )  [static]
```

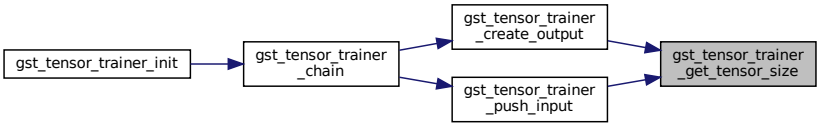
Calculate tensor buffer size.

Definition at line 1250 of file gsttensor_trainer.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.66.4.21 gst_tensor_trainer_init()

```
static void gst_tensor_trainer_init (
            GstTensorTrainer * trainer )  [static]
```

Initialize tensor_trainer.

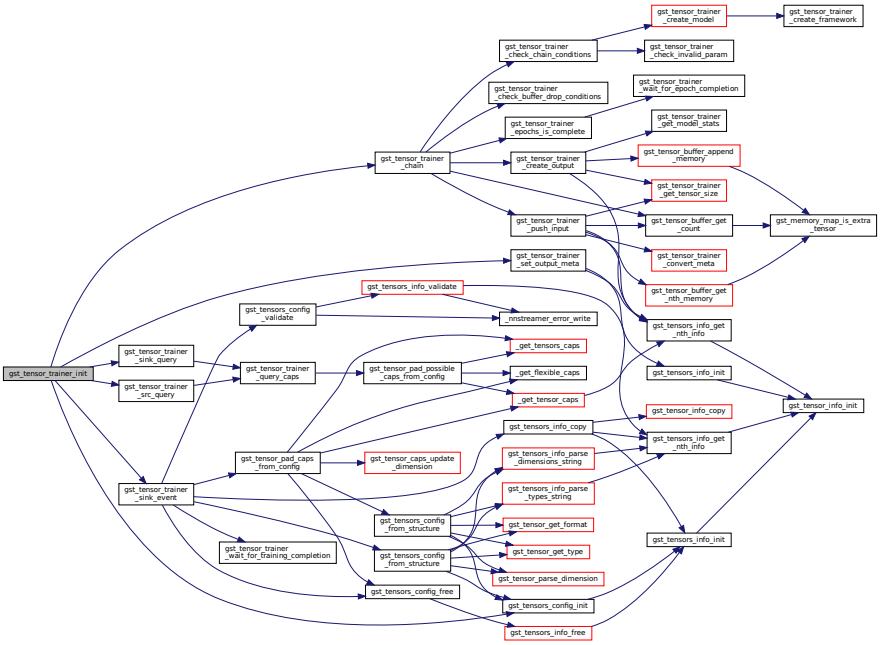
setup sink pad

setup src pad

init properties

Definition at line 262 of file gsttensor_trainer.c.

Here is the call graph for this function:



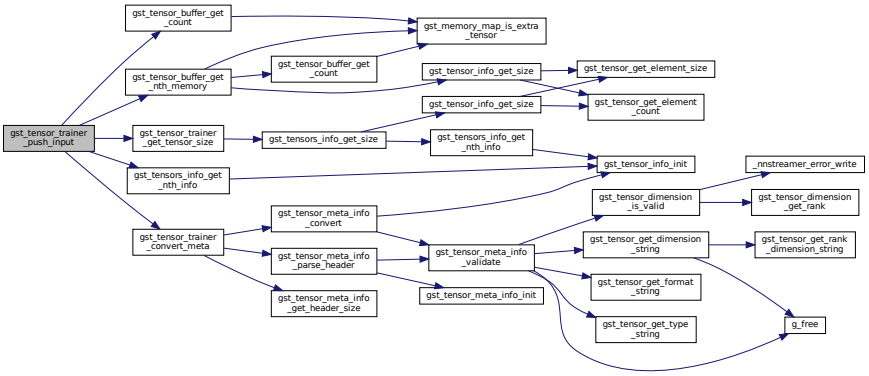
9.66.4.22 gst_tensor_trainer_push_input()

```
static gboolean gst_tensor_trainer_push_input (
            GstTensorTrainer * trainer,
            GstBuffer * inbuf,
            gboolean in_flexible )  [static]
```

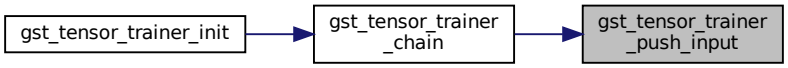
Create input tensors from the buffer and push it into trainer fw.

Definition at line 687 of file gsttensor_trainer.c.

Here is the call graph for this function:



Here is the caller graph for this function:



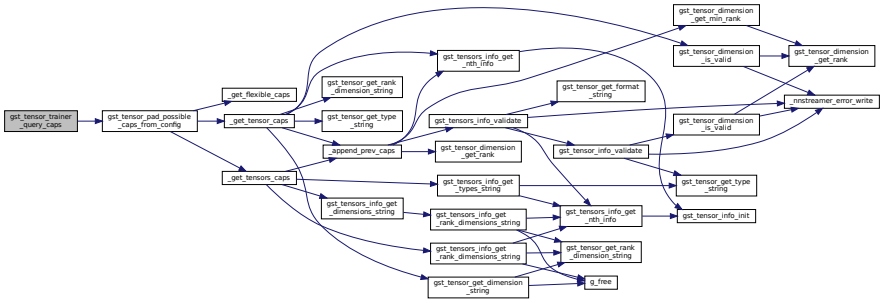
9.66.4.23 gst_tensor_trainer_query_caps()

```
static GstCaps * gst_tensor_trainer_query_caps (
            GstTensorTrainer * trainer,
            GstPad * pad,
            GstCaps * filter )  [static]
```

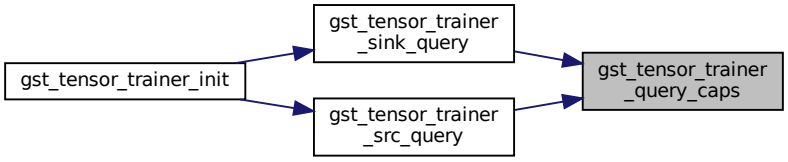
Get pad caps for caps negotiation.

Definition at line 919 of file gsttensor_trainer.c.

Here is the call graph for this function:



Here is the caller graph for this function:



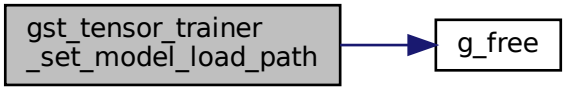
9.66.4.24 gst_tensor_trainer_set_model_load_path()

```
static void gst_tensor_trainer_set_model_load_path (
            GstTensorTrainer * trainer,
            const GValue * value )  [static]
```

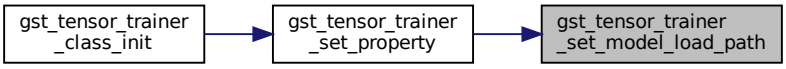
Handle "PROP_MODEL_LOAD_PATH" for set-property.

Definition at line 1183 of file gsttensor_trainer.c.

Here is the call graph for this function:



Here is the caller graph for this function:



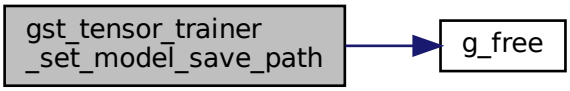
9.66.4.25 gst_tensor_trainer_set_model_save_path()

```
static void gst_tensor_trainer_set_model_save_path (
            GstTensorTrainer * trainer,
            const GValue * value )  [static]
```

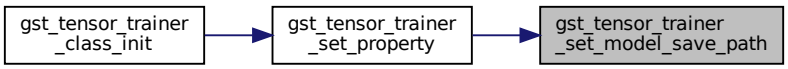
Handle "PROP_MODEL_SAVE_PATH" for set-property.

Definition at line 1170 of file gsttensor_trainer.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.66.4.26 gst_tensor_trainer_set_output_meta()

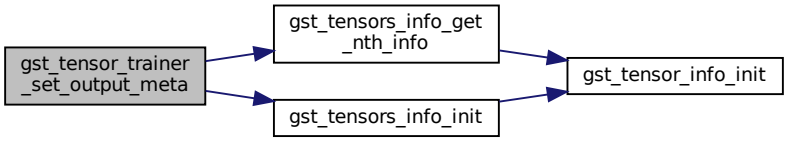
```
static void gst_tensor_trainer_set_output_meta (
             GstTensorTrainer * trainer )  [static]
```

initialize the output tensor dimension

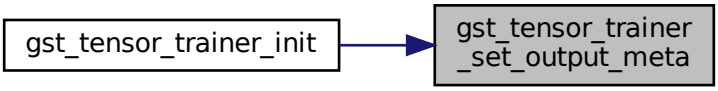
loss, accuracy, val_loss, val_accuracy

Definition at line 1347 of file gsttensor_trainer.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.66.4.27 gst_tensor_trainer_set_prop_framework()

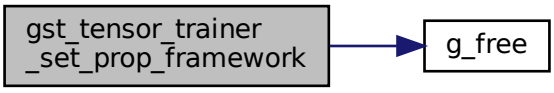
```
static void gst_tensor_trainer_set_prop_framework (
            GstTensorTrainer * trainer,
            const GValue * value )  [static]
```

Handle "PROP_FRAMEWORK" for set-property.

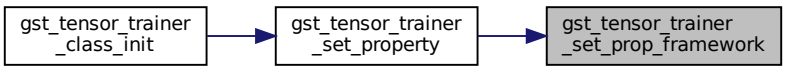
Todo Check valid framework

Definition at line 1143 of file gsttensor_trainer.c.

Here is the call graph for this function:



Here is the caller graph for this function:



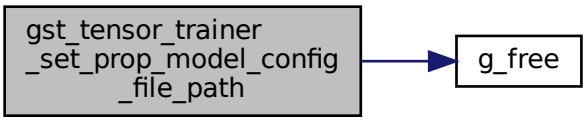
9.66.4.28 gst_tensor_trainer_set_prop_model_config_file_path()

```
static void gst_tensor_trainer_set_prop_model_config_file_path (
            GstTensorTrainer * trainer,
            const GValue * value )  [static]
```

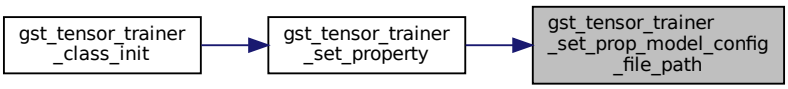
Handle "PROP_MODEL_CONFIG" for set-property.

Definition at line 1157 of file gsttensor_trainer.c.

Here is the call graph for this function:



Here is the caller graph for this function:



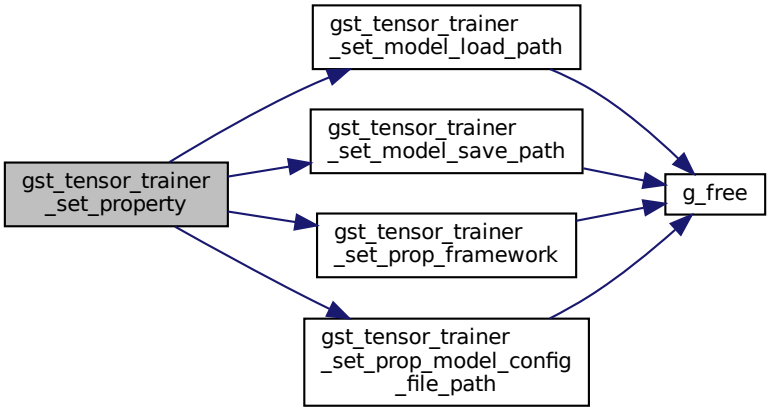
9.66.4.29 gst_tensor_trainer_set_property()

```
static void gst_tensor_trainer_set_property (
            GObject * object,
            guint prop_id,
            const GValue * value,
            GParamSpec * pspec )  [static]
```

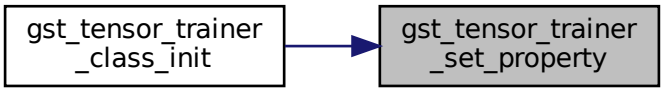
Setter for tensor_trainsink properties.

Definition at line 350 of file gsttensor_trainer.c.

Here is the call graph for this function:



Here is the caller graph for this function:



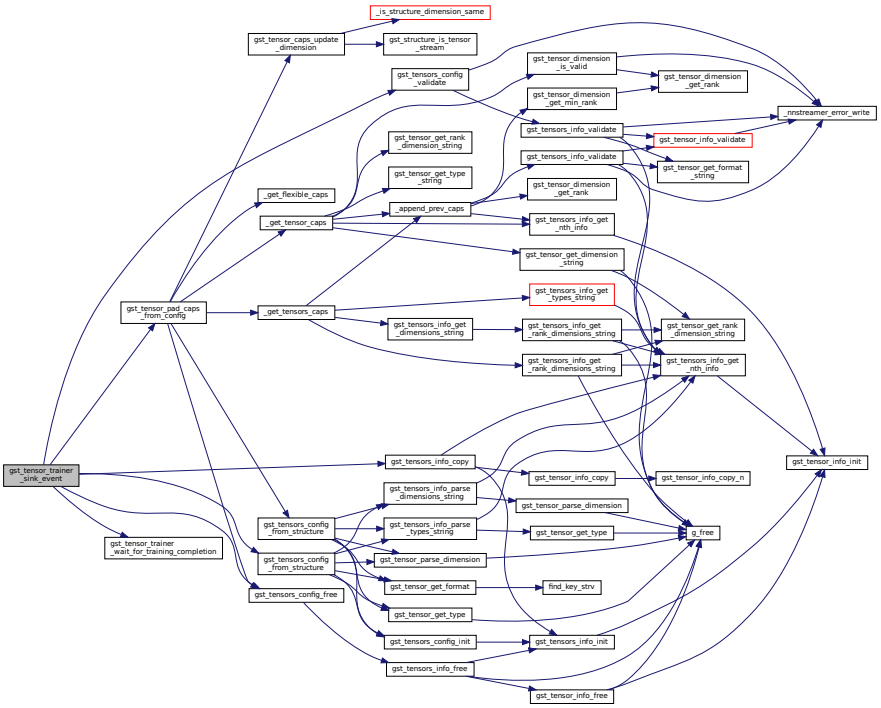
9.66.4.30 gst_tensor_trainer_sink_event()

```
static gboolean gst_tensor_trainer_sink_event (
            GstPad * sinkpad,
            GstObject * parent,
            GstEvent * event )  [static]
```

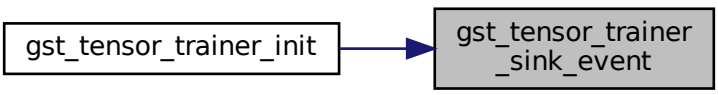
Event handler for sink pad of tensor_trainer.

Definition at line 977 of file gsttensor_trainer.c.

Here is the call graph for this function:



Here is the caller graph for this function:



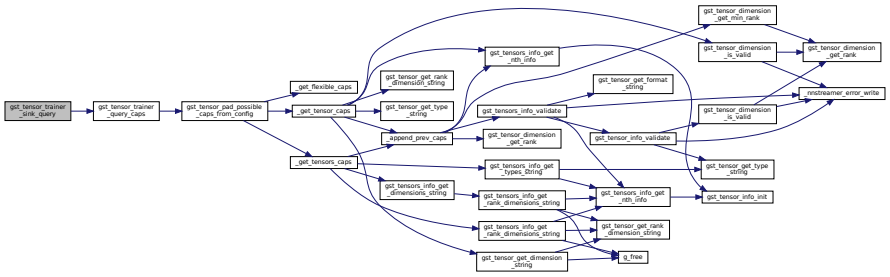
9.66.4.31 gst_tensor_trainer_sink_query()

```
static gboolean gst_tensor_trainer_sink_query (
            GstPad * sinkpad,
            GstObject * parent,
            GstQuery * query )  [static]
```

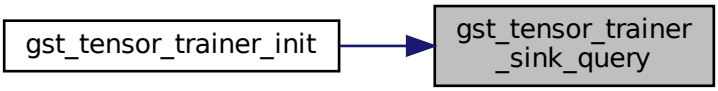
This function handles sink pad query.

Definition at line 1046 of file gsttensor_trainer.c.

Here is the call graph for this function:



Here is the caller graph for this function:



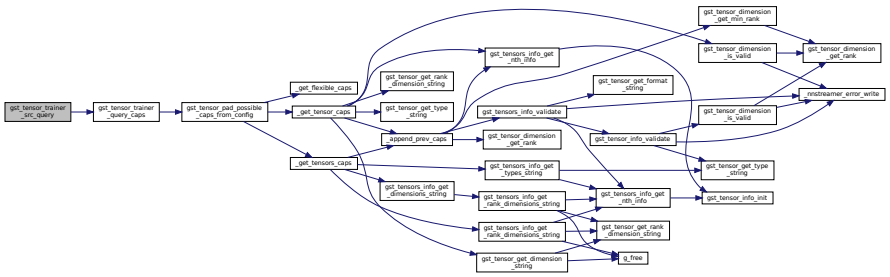
9.66.4.32 gst_tensor_trainer_src_query()

```
static gboolean gst_tensor_trainer_src_query (
            GstPad * srcpad,
            GstObject * parent,
            GstQuery * query )  [static]
```

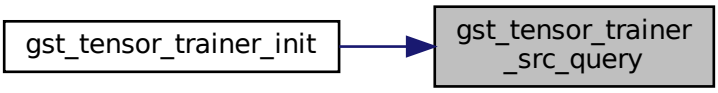
This function handles src pad query.

Definition at line 1109 of file gsttensor_trainer.c.

Here is the call graph for this function:



Here is the caller graph for this function:



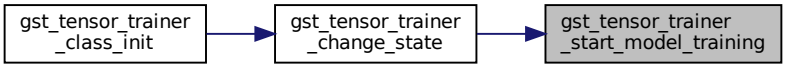
9.66.4.33 gst_tensor_trainer_start_model_training()

```
static void gst_tensor_trainer_start_model_training (
            GstTensorTrainer * trainer )  [static]
```

Start model training.

Definition at line 1308 of file gsttensor_trainer.c.

Here is the caller graph for this function:



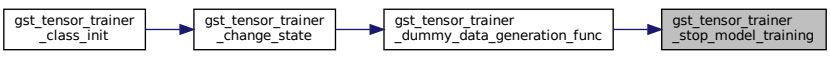
9.66.4.34 gst_tensor_trainer_stop_model_training()

```
static void gst_tensor_trainer_stop_model_training (
            GstTensorTrainer * trainer )  [static]
```

Stop model training.

Definition at line 1328 of file gsttensor_trainer.c.

Here is the caller graph for this function:



9.66.4.35 gst_tensor_trainer_wait_for_epoch_completion()

```
static void gst_tensor_trainer_wait_for_epoch_completion (
            GstTensorTrainer * trainer )  [static]
```

Wait for epoch eompletion.

Definition at line 590 of file gsttensor_trainer.c.

Here is the caller graph for this function:

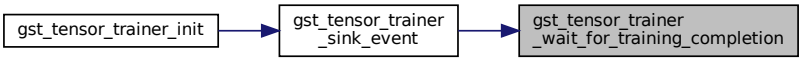
9.66.4.36 gst_tensor_trainer_wait_for_training_completion()

```
static void gst_tensor_trainer_wait_for_training_completion (
            GstTensorTrainer * trainer )  [static]
```

Wait for training completion.

Definition at line 955 of file gsttensor_trainer.c.

Here is the caller graph for this function:



9.66.4.37 nnstreamer_trainer_exit()

```
int nnstreamer_trainer_exit (
            GstTensorTrainerFramework * ttsp )
```

Trainer's sub-plugin may call this to unregister itself.

Parameters

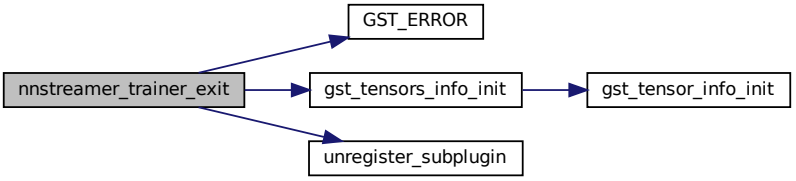
in	*ttsp*	tensor_trainer sub-plugin to be unregistered.

Returns

TRUE if unregistered. FALSE is failed.

Definition at line 1398 of file gsttensor_trainer.c.

Here is the call graph for this function:



9.66.4.38 nnstreamer_trainer_notify_event()

```
void nnstreamer_trainer_notify_event (
            GstTensorTrainerEventNotifier * notifier,
            GstTensorTrainerEventType type,
            void * data )
```

Trainer's sub-plugin may call this to send event.

Trainer's sub-plugin call this to notify event.

Parameters

in	*notifier*	event notifier, sub-plugin must send events with this.
in	*type*	event type

Definition at line 1425 of file gsttensor_trainer.c.

9.66.4.39 nnstreamer_trainer_probe()

```
int nnstreamer_trainer_probe (
            GstTensorTrainerFramework * ttsp )
```

Trainer's sub-plugin should call this function to register itself.

Parameters

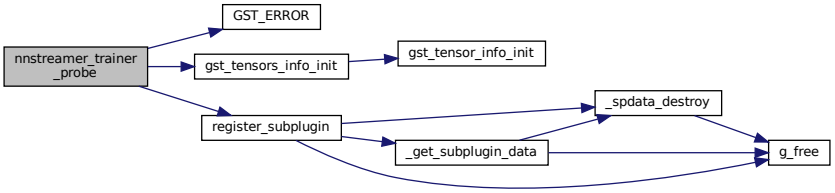
in	*ttsp*	tensor_trainer sub-plugin to be registered.

Returns

TRUE if registered. FALSE is failed or duplicated.

Definition at line 1371 of file gsttensor_trainer.c.

Here is the call graph for this function:



9.66.5 Variable Documentation

9.66.5.1 sink_template

```
GstStaticPadTemplate sink_template  [static]
```

Initial value:

```
= GST_STATIC_PAD_TEMPLATE ("sink",
    GST_PAD_SINK,
    GST_PAD_ALWAYS,
    GST_STATIC_CAPS (SINK_CAPS_STRING))
```

The capabilities of the sink pad.

Definition at line 50 of file gsttensor_trainer.c.

9.66.5.2 src_template

```
GstStaticPadTemplate src_template  [static]
```

Initial value:

```
= GST_STATIC_PAD_TEMPLATE ("src",
    GST_PAD_SRC,
    GST_PAD_ALWAYS,
    GST_STATIC_CAPS (SRC_CAPS_STRING))
```

The capabilities of the src pad.

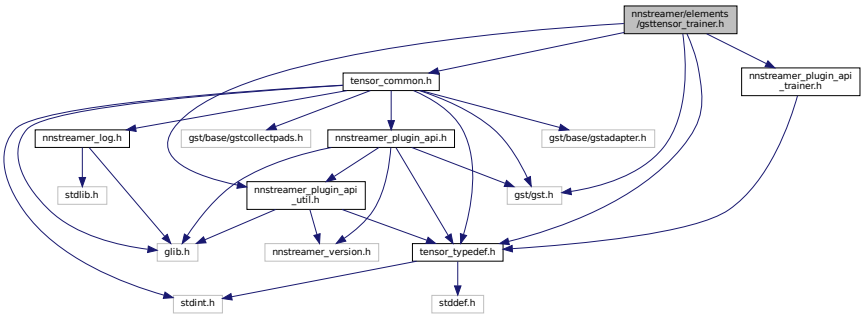
Definition at line 58 of file gsttensor_trainer.c.

9.67 nnstreamer/elements/gsttensor_trainer.h File Reference

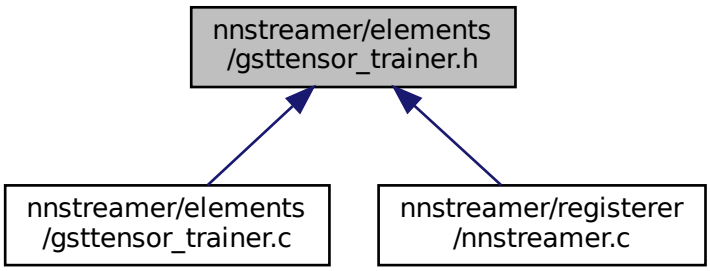
GStreamer plugin to train tensor data using NN Frameworks.

```
#include <gst/gst.h>
#include <tensor_typedef.h>
#include <tensor_common.h>
#include <nnstreamer_plugin_api_util.h>
#include <nnstreamer_plugin_api_trainer.h>
```

Include dependency graph for gsttensor_trainer.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct _GstTensorTrainer

  GstTensorTrainer data structure.
- struct _GstTensorTrainerClass

  GstTensorTrainerClass data structure.

Macros

- #define GST_TYPE_TENSOR_TRAINER (gst_tensor_trainer_get_type())
- #define GST_TENSOR_TRAINER(obj) (G_TYPE_CHECK_INSTANCE_CAST((obj),GST_TYPE_TENSOR_TRAINER,GstTens
- #define GST_TENSOR_TRAINER_CLASS(klass) (G_TYPE_CHECK_CLASS_CAST((klass),GST_TYPE_TENSOR_TRAINER
- #define GST_IS_TENSOR_TRAINER(obj) (G_TYPE_CHECK_INSTANCE_TYPE((obj),GST_TYPE_TENSOR_TRAINER))
- #define GST_IS_TENSOR_TRAINER_CLASS(klass) (G_TYPE_CHECK_CLASS_TYPE((klass),GST_TYPE_TENSOR_TRAIN

Typedefs

- typedef struct _GstTensorTrainer GstTensorTrainer
- typedef struct _GstTensorTrainerClass GstTensorTrainerClass

Functions

- GType gst_tensor_trainer_get_type (void)

  Function to get type of tensor_trainer.

9.67.1 Detailed Description

GStreamer plugin to train tensor data using NN Frameworks.

Date

20 October 2022

See also

https://github.com/nnstreamer/nnstreamer

Author

Hyunil Park hyunil46.park@samsung.com

Bug No known bugs except for NYI items

9.67.2 Macro Definition Documentation

9.67.2.1 GST_IS_TENSOR_TRAINER

```
#define GST_IS_TENSOR_TRAINER(
              obj ) (G_TYPE_CHECK_INSTANCE_TYPE((obj),GST_TYPE_TENSOR_TRAINER))
```

Definition at line 32 of file gsttensor_trainer.h.

9.67.2.2 GST_IS_TENSOR_TRAINER_CLASS

```
#define GST_IS_TENSOR_TRAINER_CLASS(
              klass ) (G_TYPE_CHECK_CLASS_TYPE((klass),GST_TYPE_TENSOR_TRAINER))
```

Definition at line 34 of file gsttensor_trainer.h.

9.67.2.3 GST_TENSOR_TRAINER

```
#define GST_TENSOR_TRAINER(
              obj ) (G_TYPE_CHECK_INSTANCE_CAST((obj),GST_TYPE_TENSOR_TRAINER,GstTensorTrainer))
```

Definition at line 28 of file gsttensor_trainer.h.

9.67.2.4 GST_TENSOR_TRAINER_CLASS

```
#define GST_TENSOR_TRAINER_CLASS(
              klass ) (G_TYPE_CHECK_CLASS_CAST((klass),GST_TYPE_TENSOR_TRAINER,GstTensorTrainerClass))
```

Definition at line 30 of file gsttensor_trainer.h.

9.67.2.5 GST_TYPE_TENSOR_TRAINER

```
#define GST_TYPE_TENSOR_TRAINER (gst_tensor_trainer_get_type())
```

Definition at line 26 of file gsttensor_trainer.h.

9.67.3 Typedef Documentation

9.67.3.1 GstTensorTrainer

```
typedef struct _GstTensorTrainer GstTensorTrainer
```

Definition at line 37 of file gsttensor_trainer.h.

9.67.3.2 GstTensorTrainerClass

```
typedef struct _GstTensorTrainerClass GstTensorTrainerClass
```

Definition at line 38 of file gsttensor_trainer.h.

9.67.4 Function Documentation

9.67.4.1 gst_tensor_trainer_get_type()

```
GType gst_tensor_trainer_get_type (
            void  )
```

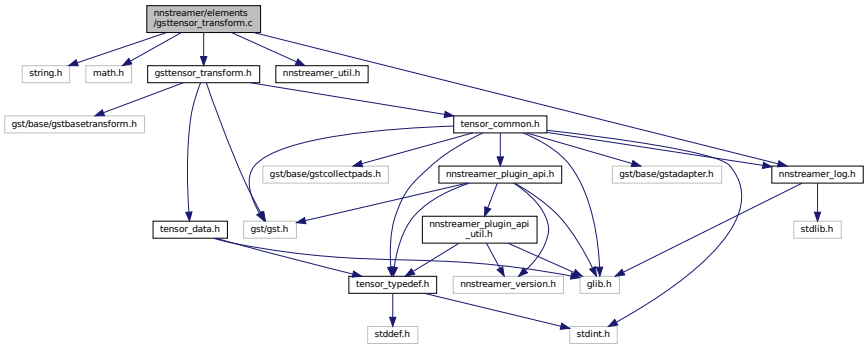
Function to get type of tensor_trainer.

9.68 nnstreamer/elements/gsttensor_transform.c File Reference

GStreamer plugin to transform tensor dimension or type.

```
#include <string.h>
#include <math.h>
#include <nnstreamer_log.h>
#include <nnstreamer_util.h>
#include "gsttensor_transform.h"
```

Include dependency graph for gsttensor_transform.c:



Macros

- #define DBG (!filter->silent)

  Macro for debug mode.
- #define GST_CAT_DEFAULT gst_tensor_transform_debug
- #define CAPS_STRING GST_TENSOR_CAP_DEFAULT ";" GST_TENSORS_CAP_MAKE ("{ static, flexible }")
- #define REGEX_DIMCHG_OPTION "^([0-9]|1[0-5]):([0-9]|1[0-5])$"
- #define REGEX_TYPECAST_OPTION "(^[u]?int(8|16|32|64)$|^float(16|32|64)$)"
- #define REGEX_TRANSPOSE_OPTION "^(?:([0-2]):(?!.*\\1)){3}3$"
- #define REGEX_STAND_OPTION "^(default|dc-average)(:([u]?int(8|16|32|64)|float(16|32|64)))?(,per-channel:(true|false))?$"
- #define REGEX_CLAMP_OPTION
- #define REGEX_PADDING_OPTION "^((left|right|top|bottom|front|back):(\\d)(,)?)+(layout:(NCHW|NH↩WC))?$"
- #define REGEX_ARITH_OPTION
- #define REGEX_ARITH_OPTION_TYPECAST "(typecast:([u]?int(8|16|32|64)|float(16|32|64)))"
- #define NNS_TENSOR_TRANSPOSE_RANK_LIMIT (4)

  The transpose rank is fixed to 4. This RANK does not affect other/tensors(s)'s NNS_TENSOR_RANK_LIMIT.
- #define NNS_TENSOR_PADDING_RANK_LIMIT (3)

The padding rank is fixed to 3. This RANK does not affect other/tensors(s)'s NNS_TENSOR_RANK_LIMIT.

- #define DEFAULT_ACCELERATION FALSE

  Flag to set orc acceleration.
- #define gst_tensor_transform_parent_class parent_class
- #define GST_TYPE_TENSOR_TRANSFORM_MODE (gst_tensor_transform_mode_get_type ())
- #define _conv_to_f16(intype, o, i, n) do { float16_not_supported (); } while (0)
- #define _conv_from_f16(otype, o, i, n) do { float16_not_supported (); } while (0)
- #define _op_float16(i, n, v, op) do { float16_not_supported (); } while (0)
- #define handle_operator(d, v, oper, vtype)

  Macro for operator.
- #define transposeloop(cl, ck, cj, ci, sl, sk, sj, si, typesize)

Enumerations

- enum {
  PROP_0, PROP_SILENT, PROP_MODE, PROP_OPTION,
  PROP_ACCELERATION, PROP_APPLY, PROP_TRANSPOSE_RANK_LIMIT }

  tensor_transform properties

Functions

- GST_DEBUG_CATEGORY_STATIC (gst_tensor_transform_debug)
- G_DEFINE_TYPE (GstTensorTransform, gst_tensor_transform, GST_TYPE_BASE_TRANSFORM)
- static void gst_tensor_transform_set_property (GObject *object, guint prop_id, const GValue *value, GParamSpec *pspec)

  Set property (gst element vmethod)
- static void gst_tensor_transform_get_property (GObject *object, guint prop_id, GValue *value, GParamSpec *pspec)

  Get property (gst element vmethod)
- static void gst_tensor_transform_finalize (GObject *object)

  Function to finalize instance (gst element vmethod)
- static GstFlowReturn gst_tensor_transform_transform (GstBaseTransform *trans, GstBuffer *inbuf, GstBuffer *outbuf)

  non-ip transform. required vmethod for BaseTransform class.
- static GstCaps * gst_tensor_transform_transform_caps (GstBaseTransform *trans, GstPadDirection direction, GstCaps *caps, GstCaps *filtercap)

  configure srcpad cap from "proposed" cap. (required vmethod for BaseTransform)
- static GstCaps * gst_tensor_transform_fixate_caps (GstBaseTransform *trans, GstPadDirection direction, GstCaps *caps, GstCaps *othercaps)

  fixate caps. required vmethod of BaseTransform
- static gboolean gst_tensor_transform_set_caps (GstBaseTransform *trans, GstCaps *incaps, GstCaps *outcaps)

  set caps. required vmethod of BaseTransform
- static gboolean gst_tensor_transform_transform_size (GstBaseTransform *trans, GstPadDirection direction, GstCaps *caps, gsize size, GstCaps *othercaps, gsize *othersize)

  Tell the framework the required size of buffer based on the info of the other side pad. Note that this is always the same with the input. optional vmethod of BaseTransform.
- static gboolean gst_tensor_transform_convert_dimension (GstTensorTransform *filter, GstPadDirection direction, guint idx, const GstTensorInfo *in_info, GstTensorInfo *out_info)

  Dimension conversion calculation.
- static GType gst_tensor_transform_mode_get_type (void)

A private function to register GEnumValue array for the 'mode' property to a GType and return it.

- static void gst_tensor_transform_class_init (GstTensorTransformClass *klass)

  initialize the tensor_transform's class
- static void gst_tensor_transform_init (GstTensorTransform *filter)

  initialize the new element (G_DEFINE_TYPE requires this) instantiate pads and add them to element set pad callback functions initialize instance structure
- static tensor_transform_operator gst_tensor_transform_get_operator (const gchar *str)

  Get the corresponding operator from the string value.
- static tensor_transform_stand_mode gst_tensor_transform_get_stand_mode (const gchar *str)

  Get the corresponding mode from the string value.
- static void float16_not_supported (void)

  Generate error if float16 is required.
- while (0)
- static gboolean gst_tensor_transform_set_option_data (GstTensorTransform *filter)

  Setup internal data (data_* in GstTensorTransform)
- static GstFlowReturn gst_tensor_transform_dimchg (GstTensorTransform *filter, GstTensorInfo *in_info, GstTensorInfo *out_info, const uint8_t *inptr, uint8_t *outptr)

  subrouting for tensor-transform, "dimchg" case.
- static GstFlowReturn gst_tensor_transform_typecast (GstTensorTransform *filter, GstTensorInfo *in_info, GstTensorInfo *out_info, const uint8_t *inptr, uint8_t *outptr)

  subrouting for tensor-transform, "typecast" case.
- static GstFlowReturn gst_tensor_transform_arithmetic (GstTensorTransform *filter, GstTensorInfo *in_info, GstTensorInfo *out_info, const uint8_t *inptr, uint8_t *outptr)

  subrouting for tensor-transform, "arithmetic" case.
- static GstFlowReturn gst_tensor_transform_transpose (GstTensorTransform *filter, GstTensorInfo *in_info, GstTensorInfo *out_info, const uint8_t *inptr, uint8_t *outptr)

  subrouting for tensor-transform, "transpose" case.
- static GstFlowReturn gst_tensor_transform_stand (GstTensorTransform *filter, GstTensorInfo *in_info, GstTensorInfo *out_info, const uint8_t *inptr, uint8_t *outptr)

  subrouting for tensor-transform, "stand" case. : pixel = abs((pixel - average(tensor))/(std(tensor) + val))
- static GstFlowReturn gst_tensor_transform_clamp (GstTensorTransform *filter, GstTensorInfo *in_info, GstTensorInfo *out_info, const uint8_t *inptr, uint8_t *outptr)

  subrouting for tensor-transform, "clamp" case. : pixel = if (pixel > max) ? max : if (pixel < min) ? min : pixel
- static GstFlowReturn gst_tensor_transform_padding (GstTensorTransform *filter, GstTensorInfo *in_info, GstTensorInfo *out_info, const uint8_t *inptr, uint8_t *outptr)

  subrouting for tensor-transform, "padding" case.
- static gboolean gst_tensor_transform_read_caps (GstTensorTransform *filter, const GstCaps *caps, GstTensorsConfig *config)

  Read cap, parse tensor configuration (dim/type) from the cap.

Variables

- static const gchar * gst_tensor_transform_stand_string [ ]
- static const gchar * gst_tensor_transform_operator_string [ ]
- static GstStaticPadTemplate sink_factory

  The capabilities of the inputs.
- static GstStaticPadTemplate src_factory

  The capabilities of the outputs.
- break

  *vtype += (v)->data._##vtype; *
- case GTT_OP_MUL
- default __pad0__
- default Unknown operator%d", oper)
- return FALSE

9.68.1 Detailed Description

GStreamer plugin to transform tensor dimension or type.

GStreamer Copyright (C) 2005 Thomas Vander Stichele thomas@apestaart.org Copyright (C) 2005 Ronald S. Bultje rbultje@ronald.bitfreak.net Copyright (C) 2018 MyungJoo Ham myungjoo.ham@samsung.com

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Library General Public License as published by the Free Software Foundation; version 2.1 of the License.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Library General Public License for more details.

Date

10 Jul 2018

See also

https://github.com/nnstreamer/nnstreamer

Author

MyungJoo Ham myungjoo.ham@samsung.com

Bug This is NYI.

9.68.2 Macro Definition Documentation

9.68.2.1 _conv_from_f16

```
#define _conv_from_f16(
              otype,
              o,
              i,
              n ) do { float16_not_supported (); } while (0)
```

Definition at line 465 of file gsttensor_transform.c.

9.68.2.2 _conv_to_f16

```
#define _conv_to_f16(
              intype,
              o,
              i,
              n ) do { float16_not_supported (); } while (0)
```

Definition at line 464 of file gsttensor_transform.c.

9.68.2.3 _op_float16

```
#define _op_float16(
              i,
              n,
              v,
              op ) do { float16_not_supported (); } while (0)
```

Definition at line 466 of file gsttensor_transform.c.

9.68.2.4 CAPS_STRING

```
#define CAPS_STRING GST_TENSOR_CAP_DEFAULT ";" GST_TENSORS_CAP_MAKE ("{ static, flexible }")
```

Definition at line 69 of file gsttensor_transform.c.

9.68.2.5 DBG

```
#define DBG (!filter->silent)
```

Macro for debug mode.

SECTION:element-tensor_transform

A filter that transforms tensors dimension or type. The input and output is always in the format of other/tensor or other/tensors.

<refsect2> <title>Example launch line</title> |[gst-launch -v -m fakesrc ! tensor_converter ! tensor_transform mode=dimchg option=0:2 ! fakesink silent=TRUE]| <title>How to use dimchg</title> |[option=0:2 # Move 0th dim to 2nd dim. I.e., [a][H][W][C] ==> [a][C][H][W]]| </refsect2>

Definition at line 64 of file gsttensor_transform.c.

9.68.2.6 DEFAULT_ACCELERATION

```
#define DEFAULT_ACCELERATION FALSE
```

Flag to set orc acceleration.

Definition at line 115 of file gsttensor_transform.c.

9.68.2.7 GST_CAT_DEFAULT

```
#define GST_CAT_DEFAULT gst_tensor_transform_debug
```

Definition at line 68 of file gsttensor_transform.c.

9.68.2.8 gst_tensor_transform_parent_class

```
#define gst_tensor_transform_parent_class parent_class
```

Definition at line 148 of file gsttensor_transform.c.

9.68.2.9 GST_TYPE_TENSOR_TRANSFORM_MODE

```
#define GST_TYPE_TENSOR_TRANSFORM_MODE (gst_tensor_transform_mode_get_type ())
```

Definition at line 176 of file gsttensor_transform.c.

9.68.2.10 handle_operator

```
#define handle_operator(
              d,
              v,
              oper,
              vtype )
```

Value:

```
    do { \
    switch (oper) { \
      case GTT_OP_ADD: \
        (d)->data._
```

Macro for operator.

Definition at line 573 of file gsttensor_transform.c.

9.68.2.11 NNS_TENSOR_PADDING_RANK_LIMIT

```
#define NNS_TENSOR_PADDING_RANK_LIMIT (3)
```

The padding rank is fixed to 3. This RANK does not affect other/tensors(s)'s NNS_TENSOR_RANK_LIMIT.

Definition at line 93 of file gsttensor_transform.c.

9.68.2.12 NNS_TENSOR_TRANSPOSE_RANK_LIMIT

```
#define NNS_TENSOR_TRANSPOSE_RANK_LIMIT (4)
```

The transpose rank is fixed to 4. This RANK does not affect other/tensors(s)'s NNS_TENSOR_RANK_LIMIT.

Definition at line 87 of file gsttensor_transform.c.

9.68.2.13 REGEX_ARITH_OPTION

```
#define REGEX_ARITH_OPTION
```

Value:

```
    "^(typecast:([u]?int(8|16|32|64)|float(16|32|64)),)?"\
    "(per-channel:(false|true@[0-9]+),)?"\
    "(((add|mul|div):([-+]?[0-9]*\\.?[0-9]+([eE][-+]?[0-9]+)?))+(@[0-9]+)?)(,|))+$"
```

Definition at line 77 of file gsttensor_transform.c.

9.68.2.14 REGEX_ARITH_OPTION_TYPECAST

```
#define REGEX_ARITH_OPTION_TYPECAST "(typecast:([u]?int(8|16|32|64)|float(16|32|64)))"
```

Definition at line 81 of file gsttensor_transform.c.

9.68.2.15 REGEX_CLAMP_OPTION

```
#define REGEX_CLAMP_OPTION
```

Value:

```
    "^((([-+]?[0-9]*\\.?[0-9]+([eE][-+]?[0-9]+)?))):"\
    "((([-+]?[0-9]*\\.?[0-9]+([eE][-+]?[0-9]+)?)))$"
```

Definition at line 74 of file gsttensor_transform.c.

9.68.2.16 REGEX_DIMCHG_OPTION

```
#define REGEX_DIMCHG_OPTION "^([0-9]|1[0-5]):([0-9]|1[0-5])$"
```

Definition at line 70 of file gsttensor_transform.c.

9.68.2.17 REGEX_PADDING_OPTION

```
#define REGEX_PADDING_OPTION "^((left|right|top|bottom|front|back):(\\d)(,)?)+(layout:(NCHW|NH↩
WC))?$"
```

Definition at line 76 of file gsttensor_transform.c.

9.68.2.18 REGEX_STAND_OPTION

```
#define REGEX_STAND_OPTION "^(default|dc-average)(:([u]?int(8|16|32|64)|float(16|32|64)))?(,per-channel↩
:(true|false))?$"
```

Definition at line 73 of file gsttensor_transform.c.

9.68.2.19 REGEX_TRANSPOSE_OPTION

```
#define REGEX_TRANSPOSE_OPTION "^(?:([0-2]):(?!.*\\1)){3}3$"
```

Definition at line 72 of file gsttensor_transform.c.

9.68.2.20 REGEX_TYPECAST_OPTION

```
#define REGEX_TYPECAST_OPTION "(^[u]?int(8|16|32|64)$|^float(16|32|64)$)"
```

Definition at line 71 of file gsttensor_transform.c.

9.68.2.21 transposeloop

```
#define transposeloop(
              cl,
              ck,
              cj,
              ci,
              sl,
              sk,
              sj,
              si,
              typesize )
```

Value:

```
    do { \
    size_t i, j, k, l;                                  \
    int inidx = 0, outidx=0;                            \
    for(cl=0;cl<sl;cl++)                    \
      for(ci=0;ci<si;ci++)                  \
        for(cj=0;cj<sj;cj++)                \
          for(ck=0;ck<sk;ck++){             \
            const uint8_t *_in; \
            uint8_t *_out; \
            outidx = si*sj*sk*cl + sj*sk*ci + sk*cj + ck; \
            inidx = SK*SJ*SI*l + SJ*SI*k + SI*j + i; \
            _in = inptr + inidx * typesize; \
            _out = outptr + outidx * typesize; \
            nns_memcpy(_out, _in, typesize); \
      }                                                 \
  } while(0);
```

Macro to run loop for various data types with transpose

Definition at line 1506 of file gsttensor_transform.c.

9.68.3 Enumeration Type Documentation

9.68.3.1 anonymous enum

```
anonymous enum
```

tensor_transform properties

Enumerator

PROP_0	
PROP_SILENT	
PROP_MODE	
PROP_OPTION	
PROP_ACCELERATION	
PROP_APPLY	
PROP_TRANSPOSE_RANK_LIMIT	

Definition at line 98 of file gsttensor_transform.c.

9.68.4 Function Documentation

9.68.4.1 float16_not_supported()

```
static void float16_not_supported (
            void  )  [static]
```

Generate error if float16 is required.

Definition at line 348 of file gsttensor_transform.c.

Here is the call graph for this function:

Here is the caller graph for this function:

9.68.4.2 G_DEFINE_TYPE()

```
G_DEFINE_TYPE (
             GstTensorTransform ,
             gst_tensor_transform ,
             GST_TYPE_BASE_TRANSFORM  )
```

9.68.4.3 GST_DEBUG_CATEGORY_STATIC()

```
GST_DEBUG_CATEGORY_STATIC (
             gst_tensor_transform_debug  )
```

9.68.4.4 gst_tensor_transform_arithmetic()

```
static GstFlowReturn gst_tensor_transform_arithmetic (
             GstTensorTransform * filter,
             GstTensorInfo * in_info,
             GstTensorInfo * out_info,
             const uint8_t * inptr,
             uint8_t * outptr )  [static]
```

subrouting for tensor-transform, "arithmetic" case.

Parameters

	[in/out]	filter "this" pointer
in	*in_info*	input tensor info
in	*out_info*	output tensor info
in	*inptr*	input tensor
out	*outptr*	output tensor

Returns

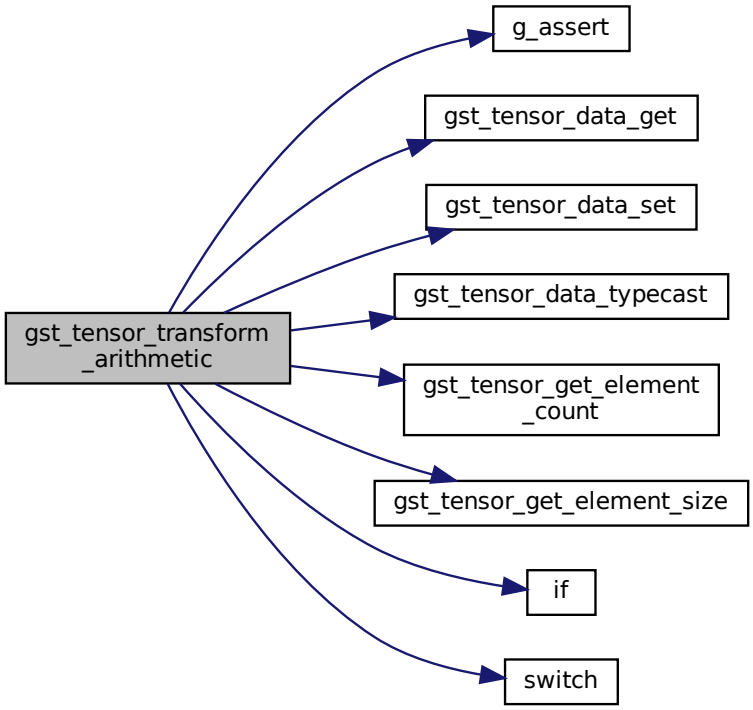
Gst flow status

In case of 3:4:4:1, ch_dim:0 -> #ch: 3, ch_size: 1, ch_offset: 3 ch_dim:1 -> #ch: 4, ch_size: 3, ch_offset: 12 ch_dim:2 -> #ch: 4, ch_size: 12, ch_offset: 48 ch_dim:3 -> #ch: 1, ch_size: 48, ch_offset: 48 * 4

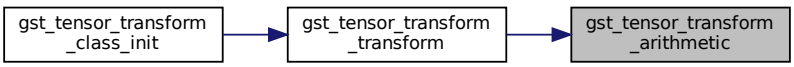
Todo add more options

Definition at line 1330 of file gsttensor_transform.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.68.4.5 gst_tensor_transform_clamp()

```
static GstFlowReturn gst_tensor_transform_clamp (
            GstTensorTransform * filter,
            GstTensorInfo * in_info,
            GstTensorInfo * out_info,
            const uint8_t * inptr,
            uint8_t * outptr )  [static]
```

subrouting for tensor-transform, "clamp" case. : pixel = if (pixel > max) ? max : if (pixel < min) ? min : pixel

Parameters

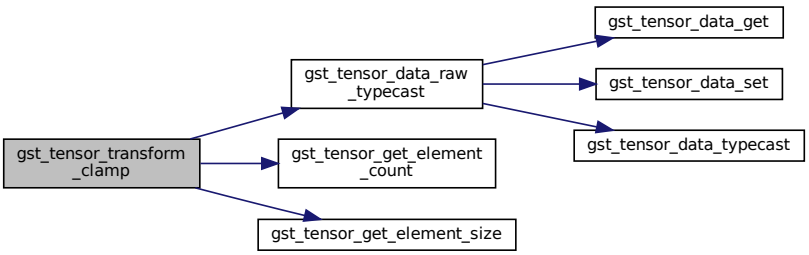
	[in/out]	filter "this" pointer
in	*in_info*	input tensor info
in	*out_info*	output tensor info
in	*inptr*	input tensor
out	*outptr*	output tensor

Returns

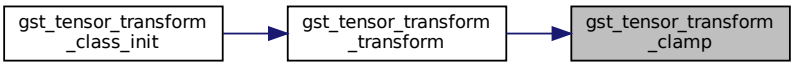
Gst flow status

Definition at line 1725 of file gsttensor_transform.c.

Here is the call graph for this function:



Here is the caller graph for this function:



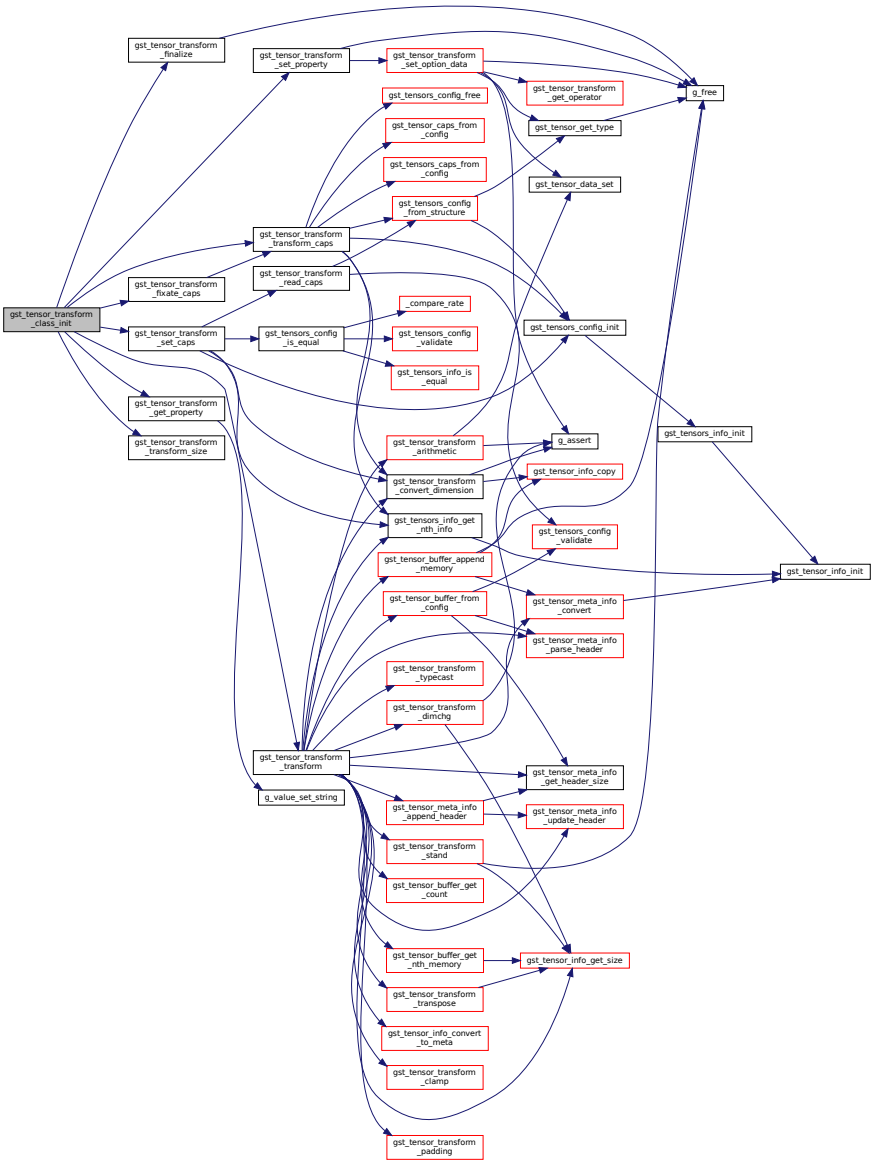
9.68.4.6 gst_tensor_transform_class_init()

```
static void gst_tensor_transform_class_init (
            GstTensorTransformClass * klass )  [static]
```

initialize the tensor_transform's class

Definition at line 224 of file gsttensor_transform.c.

Here is the call graph for this function:



9.68.4.7 gst_tensor_transform_convert_dimension()

```
static gboolean gst_tensor_transform_convert_dimension (
            GstTensorTransform * filter,
            GstPadDirection direction,
            guint idx,
            const GstTensorInfo * in_info,
            GstTensorInfo * out_info )  [static]
```

Dimension conversion calculation.

Parameters

in	*filter*	"this" pointer
in	*direction*	GST_PAD_SINK if input->output conv
in	*idx*	index of the input tensors
in	*in_info*	tensor info structure of source tensor (input if direction is SINK)
out	*out_info*	tensor info structure of destination tensor (output if direction is SINK)

Returns

TRUE if success

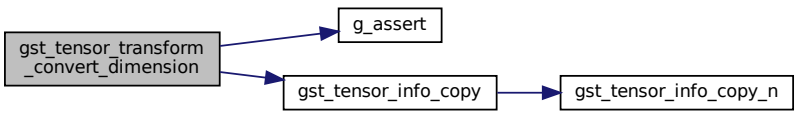
For both directions, dimension does not change

src = SINKPAD / dest = SRCPAD

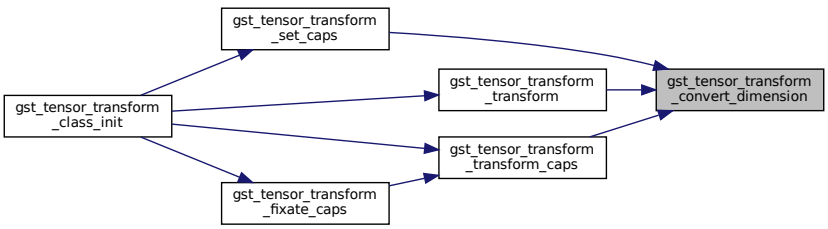
For both directions, dimension does not change

Definition at line 2005 of file gsttensor_transform.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.68.4.8 gst_tensor_transform_dimchg()

```
static GstFlowReturn gst_tensor_transform_dimchg (
            GstTensorTransform * filter,
            GstTensorInfo * in_info,
            GstTensorInfo * out_info,
            const uint8_t * inptr,
            uint8_t * outptr )  [static]
```

subrouting for tensor-transform, "dimchg" case.

Parameters

	[in/out]	filter "this" pointer
in	*in_info*	input tensor info
in	*out_info*	output tensor info
in	*inptr*	input tensor
out	*outptr*	output tensor

Returns

Gst flow status

Useless memcpy. Do not call this or

Todo do "IP" operation

Smaller-loop-ed a to larger-loop-ed b E.g., [N][H][W][c] (c:W:H:N) --> [N][c][H][W] (W:H:c:N)

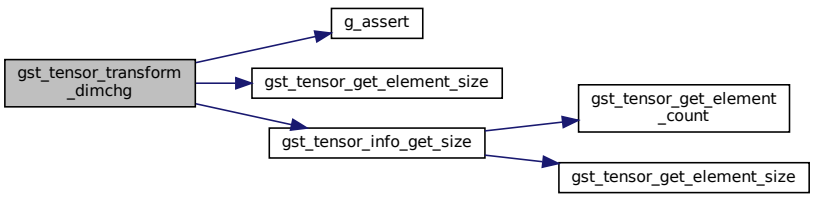
Todo CRITICAL-TODO: Optimize the performance!

Larger-loop-ed a to smaller-loop-ed b E.g., [N][c][H][W] (W:H:c:N) --> [N][H][W][c] (c:W:H:N)

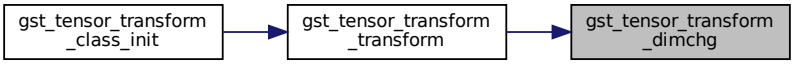
Todo NYI

Definition at line 1197 of file gsttensor_transform.c.

Here is the call graph for this function:



Here is the caller graph for this function:



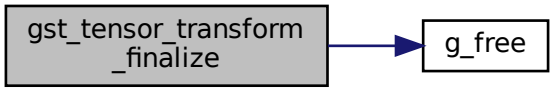
9.68.4.9 gst_tensor_transform_finalize()

```
static void gst_tensor_transform_finalize (
            GObject * object )  [static]
```

Function to finalize instance (gst element vmethod)

Definition at line 1163 of file gsttensor_transform.c.

Here is the call graph for this function:



Here is the caller graph for this function:

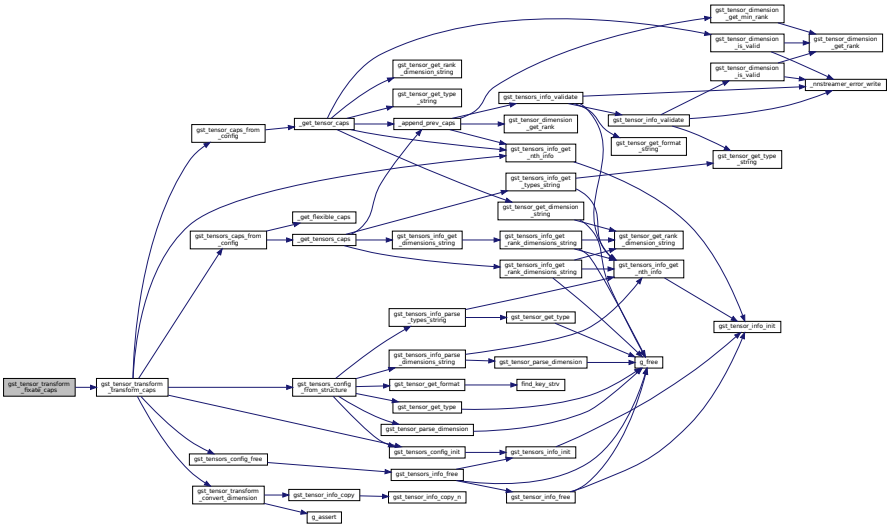
9.68.4.10 gst_tensor_transform_fixate_caps()

```
static GstCaps * gst_tensor_transform_fixate_caps (
            GstBaseTransform * trans,
            GstPadDirection direction,
            GstCaps * caps,
            GstCaps * othercaps )  [static]
```

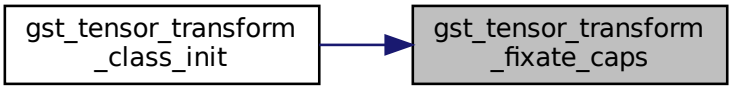
fixate caps. required vmethod of BaseTransform

Definition at line 2222 of file gsttensor_transform.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.68.4.11 gst_tensor_transform_get_operator()

```
static tensor_transform_operator gst_tensor_transform_get_operator (
            const gchar * str )  [static]
```

Get the corresponding operator from the string value.

Parameters

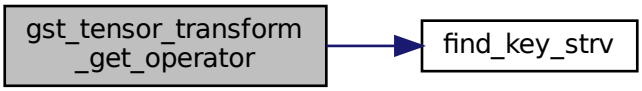
in	*str*	The string value for the operator

Returns

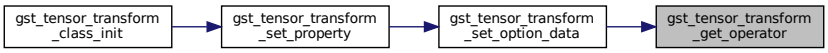
corresponding operator for the string (GTT_OP_UNKNOWN for errors)

Definition at line 319 of file gsttensor_transform.c.

Here is the call graph for this function:



Here is the caller graph for this function:



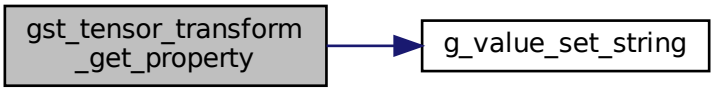
9.68.4.12 gst_tensor_transform_get_property()

```
static void gst_tensor_transform_get_property (
            GObject * object,
            guint prop_id,
            GValue * value,
            GParamSpec * pspec )  [static]
```

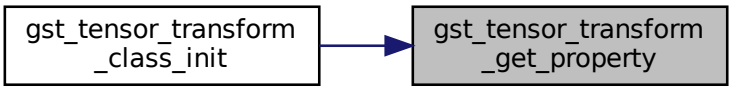
Get property (gst element vmethod)

Definition at line 1107 of file gsttensor_transform.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.68.4.13 gst_tensor_transform_get_stand_mode()

```
static tensor_transform_stand_mode gst_tensor_transform_get_stand_mode (
            const gchar * str )  [static]
```

Get the corresponding mode from the string value.

Parameters

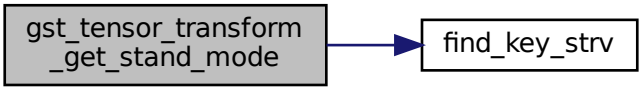
in	*str*	The string value for the mode

Returns

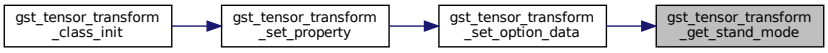
corresponding mode for the string. STAND_END for errors

Definition at line 334 of file gsttensor_transform.c.

Here is the call graph for this function:



Here is the caller graph for this function:



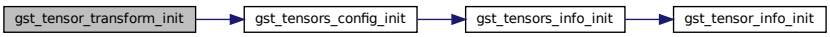
9.68.4.14 gst_tensor_transform_init()

```
static void gst_tensor_transform_init (
            GstTensorTransform * filter )  [static]
```

initialize the new element (G_DEFINE_TYPE requires this) instantiate pads and add them to element set pad callback functions initialize instance structure

Definition at line 299 of file gsttensor_transform.c.

Here is the call graph for this function:



9.68.4.15 gst_tensor_transform_mode_get_type()

```
static GType gst_tensor_transform_mode_get_type (
            void  )  [static]
```

A private function to register GEnumValue array for the 'mode' property to a GType and return it.

Definition at line 182 of file gsttensor_transform.c.

9.68.4.16 gst_tensor_transform_padding()

```
static GstFlowReturn gst_tensor_transform_padding (
            GstTensorTransform * filter,
            GstTensorInfo * in_info,
            GstTensorInfo * out_info,
            const uint8_t * inptr,
            uint8_t * outptr )  [static]
```

subrouting for tensor-transform, "padding" case.

Parameters

	[in/out]	filter "this" pointer
in	*in_info*	input tensor info
in	*out_info*	output tensor info
in	*inptr*	input tensor
out	*outptr*	output tensor

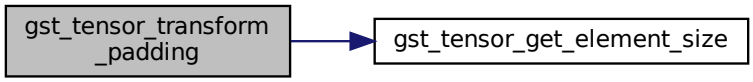
Returns

Gst flow status

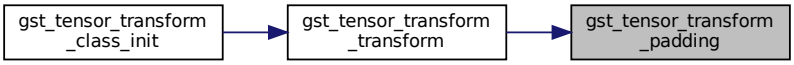
Todo Add constant option instead of using zero padding value

Definition at line 1762 of file gsttensor_transform.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.68.4.17 gst_tensor_transform_read_caps()

```
static gboolean gst_tensor_transform_read_caps (
            GstTensorTransform * filter,
            const GstCaps * caps,
            GstTensorsConfig * config )  [static]
```

Read cap, parse tensor configuration (dim/type) from the cap.

Parameters

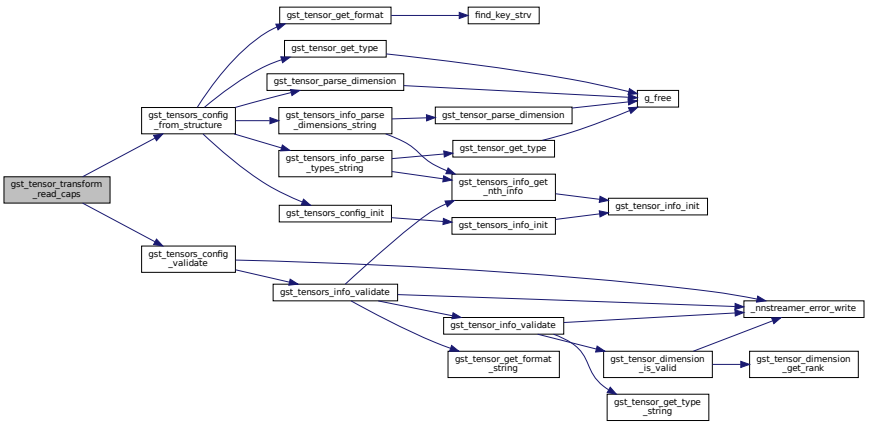
in	*filter*	"this" pointer
in	*caps*	The input caps to be read
out	*config*	configured tensor info

Returns

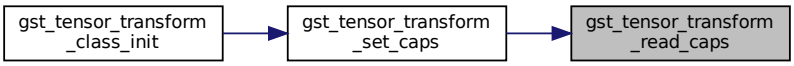
TRUE if successful (both dim/type read). FALSE if not.

Definition at line 1978 of file gsttensor_transform.c.

Here is the call graph for this function:



Here is the caller graph for this function:



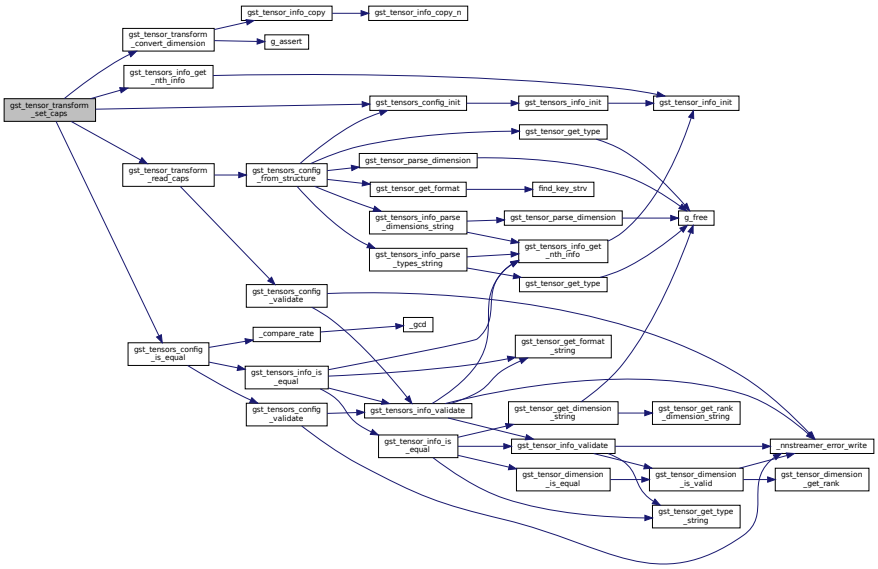
9.68.4.18 gst_tensor_transform_set_caps()

```
static gboolean gst_tensor_transform_set_caps (
            GstBaseTransform * trans,
            GstCaps * incaps,
            GstCaps * outcaps )  [static]
```

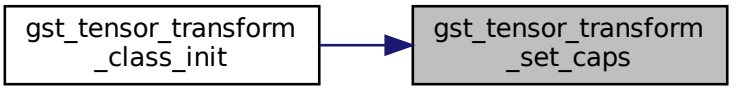
set caps. required vmethod of BaseTransform

Definition at line 2249 of file gsttensor_transform.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.68.4.19 gst_tensor_transform_set_option_data()

```
static gboolean gst_tensor_transform_set_option_data (
            GstTensorTransform * filter )  [static]
```

Setup internal data (data_* in GstTensorTransform)

Parameters

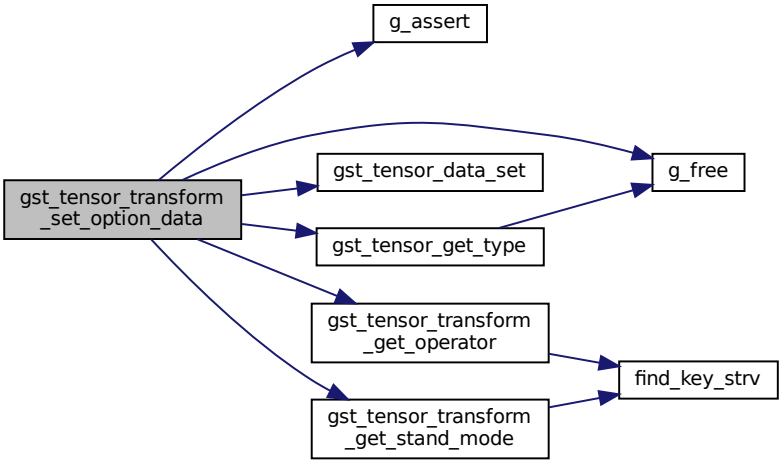
[in/out]	filter "this" pointer. mode & option MUST BE set already.

Return values

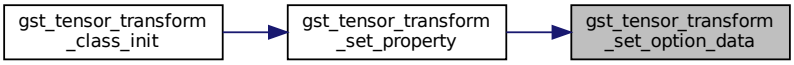
TRUE	if OK or operation-skipped, FALSE if fatal-error.

Definition at line 663 of file gsttensor_transform.c.

Here is the call graph for this function:



Here is the caller graph for this function:



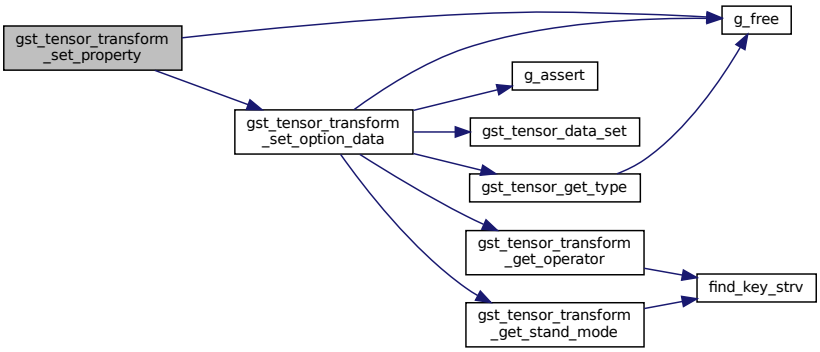
9.68.4.20 gst_tensor_transform_set_property()

```
static void gst_tensor_transform_set_property (
            GObject * object,
            guint prop_id,
            const GValue * value,
            GParamSpec * pspec )  [static]
```

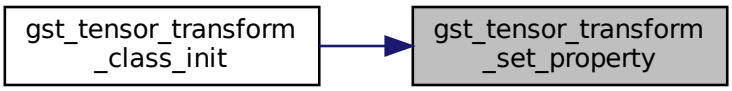
Set property (gst element vmethod)

Definition at line 1040 of file gsttensor_transform.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.68.4.21 gst_tensor_transform_stand()

```
static GstFlowReturn gst_tensor_transform_stand (
            GstTensorTransform * filter,
            GstTensorInfo * in_info,
            GstTensorInfo * out_info,
            const uint8_t * inptr,
            uint8_t * outptr )  [static]
```

subrouting for tensor-transform, "stand" case. : pixel = abs((pixel - average(tensor))/(std(tensor) + val))

Parameters

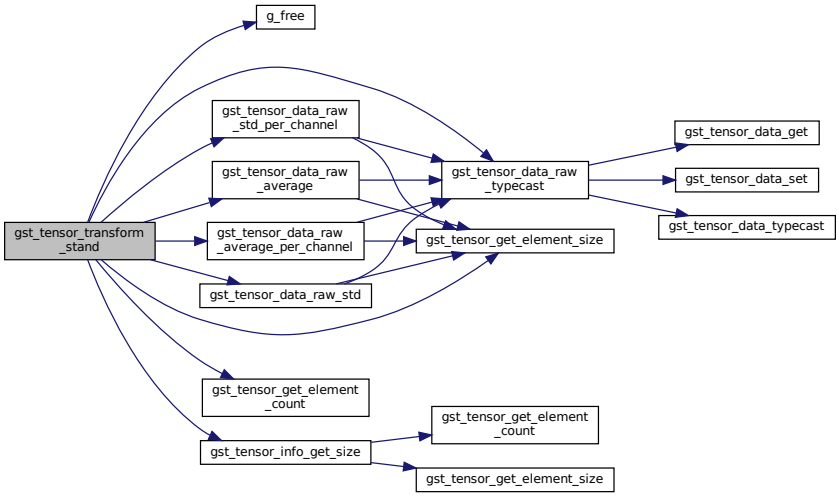
	[in/out]	filter "this" pointer
in	*in_info*	input tensor info
in	*out_info*	output tensor info
in	*inptr*	input tensor
out	*outptr*	output tensor

Returns

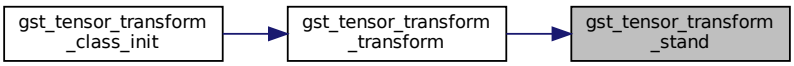
Gst flow status

Definition at line 1605 of file gsttensor_transform.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.68.4.22 gst_tensor_transform_transform()

```
static GstFlowReturn gst_tensor_transform_transform (
            GstBaseTransform * trans,
            GstBuffer * inbuf,
            GstBuffer * outbuf )  [static]
```

non-ip transform. required vmethod for BaseTransform class.

Parameters

	[in/out]	trans "super" pointer
in	*inbuf*	The input gst buffer
out	*outbuf*	The output gst buffer

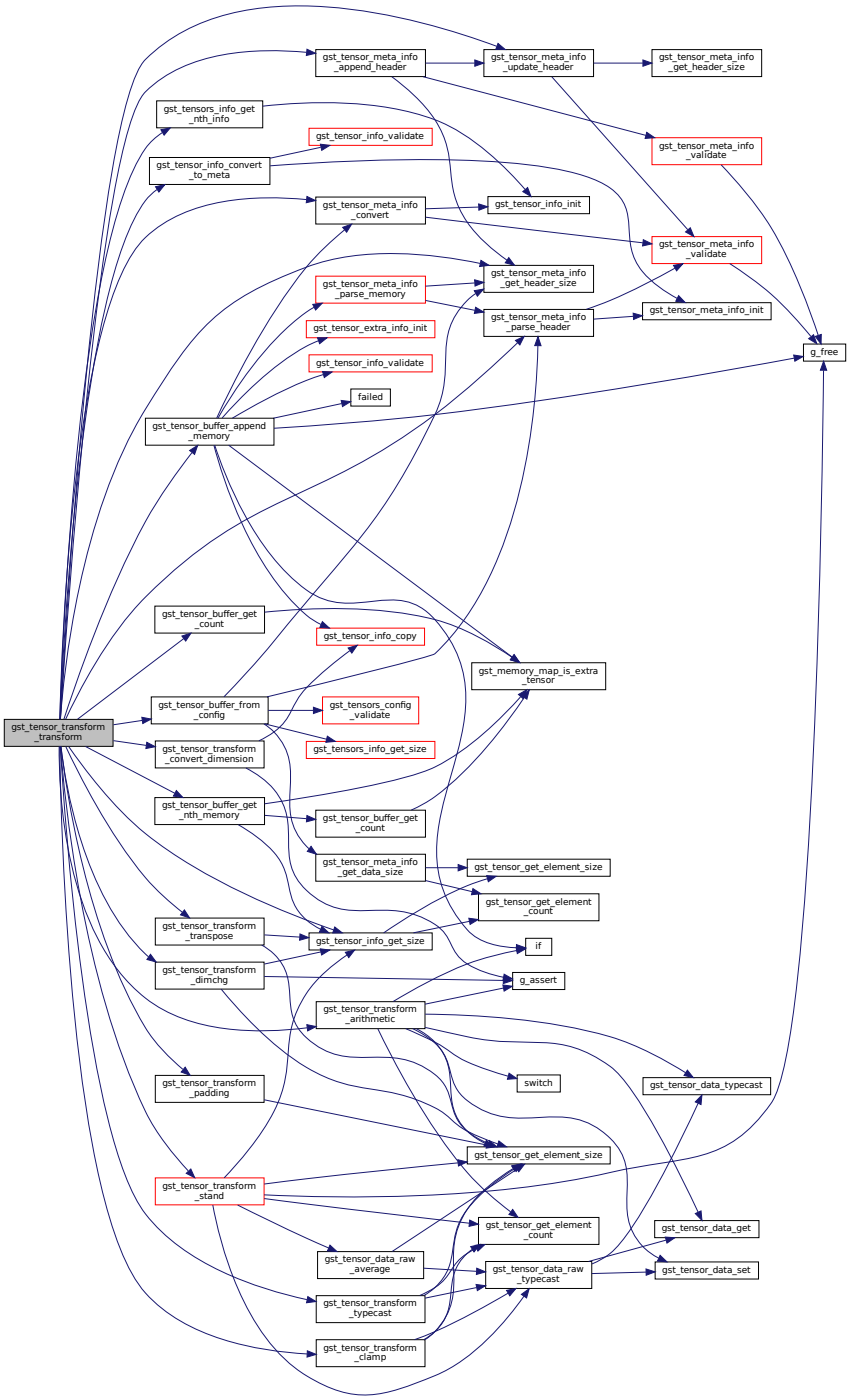
Returns

Gst Flow Status

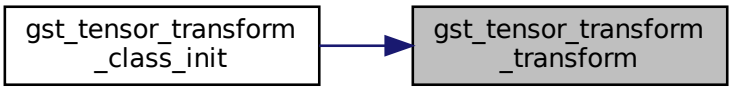
Todo max rank supported in tensor-transform is 4

Definition at line 1815 of file gsttensor_transform.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.68.4.23 gst_tensor_transform_transform_caps()

```
static GstCaps * gst_tensor_transform_transform_caps (
            GstBaseTransform * trans,
            GstPadDirection direction,
            GstCaps * caps,
            GstCaps * filtercap )  [static]
```

configure srcpad cap from "proposed" cap. (required vmethod for BaseTransform)

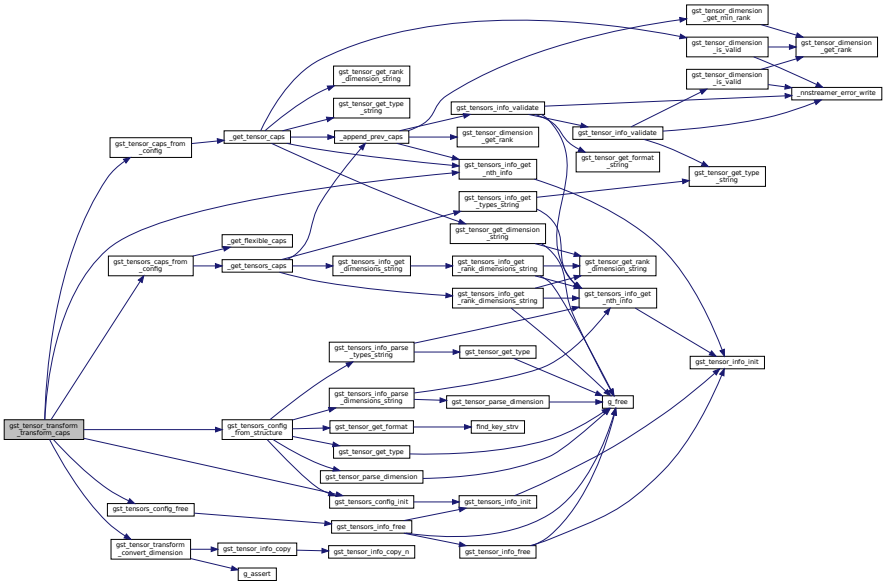
Parameters

trans	("this" pointer)
direction	(why do we need this?)
caps	sinkpad cap
filtercap	this element's cap (don't know specifically.)

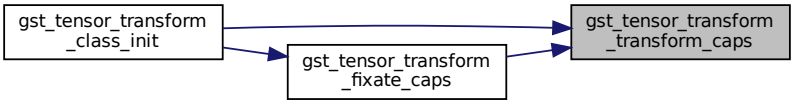
Todo Get to know what the heck is @filtercap and use it!

Definition at line 2139 of file gsttensor_transform.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.68.4.24 gst_tensor_transform_transform_size()

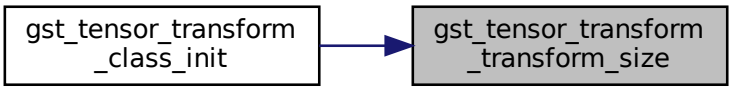
```
static gboolean gst_tensor_transform_transform_size (
            GstBaseTransform * trans,
            GstPadDirection direction,
            GstCaps * caps,
            gsize size,
            GstCaps * othercaps,
            gsize * othersize )  [static]
```

Tell the framework the required size of buffer based on the info of the other side pad. Note that this is always the same with the input. optional vmethod of BaseTransform.

Consider multi-tensors. Set each memory block in transform()

Definition at line 2329 of file gsttensor_transform.c.

Here is the caller graph for this function:



9.68.4.25 gst_tensor_transform_transpose()

```
static GstFlowReturn gst_tensor_transform_transpose (
            GstTensorTransform * filter,
            GstTensorInfo * in_info,
            GstTensorInfo * out_info,
            const uint8_t * inptr,
            uint8_t * outptr )  [static]
```

subrouting for tensor-transform, "transpose" case.

Parameters

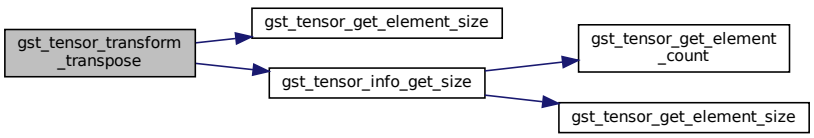
	[in/out]	filter "this" pointer
in	*in_info*	input tensor info
in	*out_info*	output tensor info
in	*inptr*	input tensor
out	*outptr*	output tensor

Returns

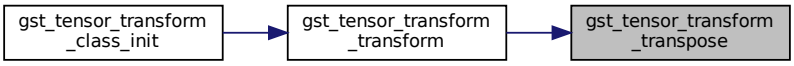
Gst flow status

Definition at line 1533 of file gsttensor_transform.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.68.4.26 gst_tensor_transform_typecast()

```
static GstFlowReturn gst_tensor_transform_typecast (
            GstTensorTransform * filter,
            GstTensorInfo * in_info,
            GstTensorInfo * out_info,
            const uint8_t * inptr,
            uint8_t * outptr )  [static]
```

subrouting for tensor-transform, "typecast" case.

Parameters

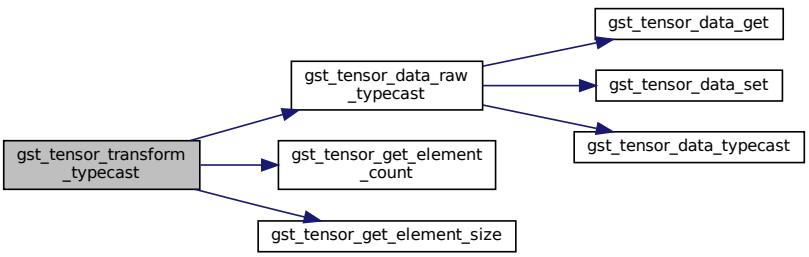
	[in/out]	filter "this" pointer
in	*in_info*	input tensor info
in	*out_info*	output tensor info
in	*inptr*	input tensor
out	*outptr*	output tensor

Returns

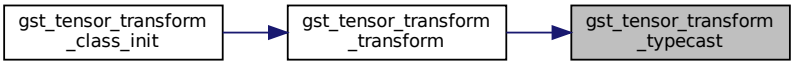
Gst flow status

Definition at line 1292 of file gsttensor_transform.c.

Here is the call graph for this function:



Here is the caller graph for this function:

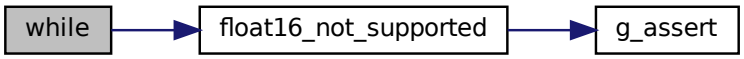


9.68.4.27 while()

```
while (
             0  )
```

Definition at line 592 of file gsttensor_transform.c.

Here is the call graph for this function:



9.68.5 Variable Documentation

9.68.5.1 __pad0__

```
default __pad0__
```

Definition at line 589 of file gsttensor_transform.c.

9.68.5.2 break

```
d data _ break
```

vtype += (v)->data._##vtype; \

vtype /= (v)->data._##vtype; \

Definition at line 577 of file gsttensor_transform.c.

9.68.5.3 FALSE

```
return FALSE
```

Definition at line 590 of file gsttensor_transform.c.

9.68.5.4 gst_tensor_transform_operator_string

```
const gchar* gst_tensor_transform_operator_string[]  [static]
```

Initial value:

```
= {
  [GTT_OP_TYPECAST] = "typecast",
  [GTT_OP_ADD] = "add",
  [GTT_OP_MUL] = "mul",
  [GTT_OP_DIV] = "div",
  [GTT_OP_UNKNOWN] = NULL
}
```

Definition at line 124 of file gsttensor_transform.c.

9.68.5.5 gst_tensor_transform_stand_string

```
const gchar* gst_tensor_transform_stand_string[]  [static]
```

Initial value:

```
= {
  [STAND_DEFAULT] = "default",
  [STAND_DC_AVERAGE] = "dc-average",
  [STAND_END] = NULL
}
```

Definition at line 118 of file gsttensor_transform.c.

9.68.5.6 GTT_OP_MUL

```
case GTT_OP_MUL
```

Definition at line 584 of file gsttensor_transform.c.

9.68.5.7 operator%d"", oper)

```
default Unknown operator%d", oper)
```

Definition at line 589 of file gsttensor_transform.c.

9.68.5.8 sink_factory

```
GstStaticPadTemplate sink_factory  [static]
```

Initial value:

```
= GST_STATIC_PAD_TEMPLATE ("sink",
    GST_PAD_SINK,
    GST_PAD_ALWAYS,
    GST_STATIC_CAPS (CAPS_STRING))
```

The capabilities of the inputs.

Definition at line 135 of file gsttensor_transform.c.

9.68.5.9 src_factory

```
GstStaticPadTemplate src_factory  [static]
```

Initial value:

```
= GST_STATIC_PAD_TEMPLATE ("src",
    GST_PAD_SRC,
    GST_PAD_ALWAYS,
    GST_STATIC_CAPS (CAPS_STRING))
```

The capabilities of the outputs.

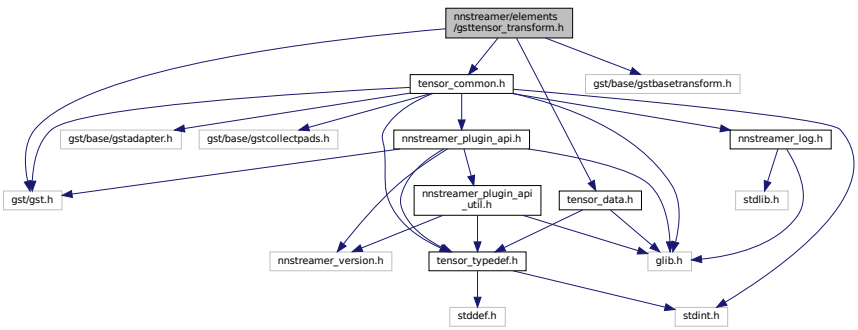
Definition at line 143 of file gsttensor_transform.c.

9.69 nnstreamer/elements/gsttensor_transform.h File Reference

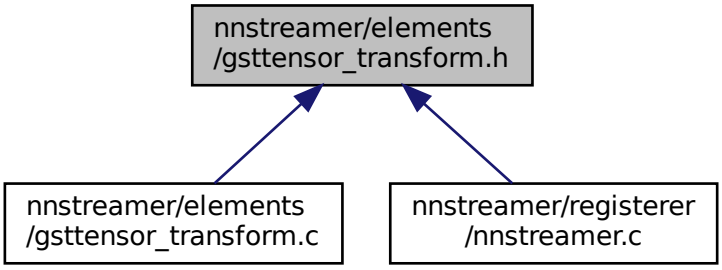
GStreamer plugin to transform tensor dimension or type.

```
#include <gst/gst.h>
#include <gst/base/gstbasetransform.h>
#include <tensor_common.h>
#include <tensor_data.h>
```

Include dependency graph for gsttensor_transform.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct _tensor_transform_dimchg

  Internal data structure for dimchg mode.
- struct _tensor_transform_typecast

  Internal data structure for typecast mode.
- struct tensor_transform_operator_s

  Internal data structure for operator of arithmetic mode.
- struct _tensor_transform_arithmetic

  Internal data structure for arithmetic mode.
- struct _tensor_transform_transpose

  Internal data structure for transpose mode.
- struct _tensor_transform_stand

  Internal data structure for stand mode.
- struct _tensor_transform_clamp

  Internal data structure for clamp mode.
- struct _tensor_transform_padding

  Internal data structure for padding mode.
- struct _GstTensorTransform

  Internal data structure for tensor_transform instances.
- struct _GstTensorTransformClass

  GstTensorTransformClass inherits GstBaseTransformClass.

Macros

- #define GST_TYPE_TENSOR_TRANSFORM (gst_tensor_transform_get_type())
- #define GST_TENSOR_TRANSFORM(obj) (G_TYPE_CHECK_INSTANCE_CAST((obj),GST_TYPE_TENSOR_TRANSFORM
- #define GST_TENSOR_TRANSFORM_CLASS(klass) (G_TYPE_CHECK_CLASS_CAST((klass),GST_TYPE_TENSOR_TRA
- #define GST_IS_TENSOR_TRANSFORM(obj) (G_TYPE_CHECK_INSTANCE_TYPE((obj),GST_TYPE_TENSOR_TRANSFO
- #define GST_IS_TENSOR_TRANSFORM_CLASS(klass) (G_TYPE_CHECK_CLASS_TYPE((klass),GST_TYPE_TENSOR_T
- #define GST_TENSOR_TRANSFORM_CAST(obj) ((GstTensorTransform *)(obj))

Typedefs

- typedef struct _GstTensorTransform GstTensorTransform
- typedef struct _GstTensorTransformClass GstTensorTransformClass
- typedef enum _tensor_transform_mode tensor_transform_mode
- typedef struct _tensor_transform_dimchg tensor_transform_dimchg

  Internal data structure for dimchg mode.
- typedef struct _tensor_transform_typecast tensor_transform_typecast

  Internal data structure for typecast mode.
- typedef struct _tensor_transform_arithmetic tensor_transform_arithmetic

  Internal data structure for arithmetic mode.
- typedef struct _tensor_transform_transpose tensor_transform_transpose

  Internal data structure for transpose mode.
- typedef struct _tensor_transform_stand tensor_transform_stand

  Internal data structure for stand mode.
- typedef struct _tensor_transform_clamp tensor_transform_clamp

  Internal data structure for clamp mode.
- typedef struct _tensor_transform_padding tensor_transform_padding

  Internal data structure for padding mode.

Enumerations

- enum _tensor_transform_mode {
  GTT_DIMCHG = 0, GTT_TYPECAST, GTT_ARITHMETIC, GTT_TRANSPOSE,
  GTT_STAND, GTT_CLAMP, GTT_PADDING, GTT_UNKNOWN = -1 }
- enum tensor_transform_operator {
  GTT_OP_TYPECAST = 0, GTT_OP_ADD = 1, GTT_OP_MUL = 2, GTT_OP_DIV = 3,
  GTT_OP_UNKNOWN }
- enum tensor_transform_stand_mode { STAND_DEFAULT = 0, STAND_DC_AVERAGE = 1, STAND_END }
- enum tensor_transform_padding_axis {
  PADDING_LEFT = 0, PADDING_RIGHT = 1, PADDING_TOP = 2, PADDING_BOTTOM = 3,
  PADDING_FRONT = 4, PADDING_BACK = 5, PADDING_END }

Functions

- GType gst_tensor_transform_get_type (void)

  Get Type function required for gst elements.

9.69.1 Detailed Description

GStreamer plugin to transform tensor dimension or type.

GStreamer Copyright (C) 2005 Thomas Vander Stichele thomas@apestaart.org Copyright (C) 2005 Ronald S. Bultje rbultje@ronald.bitfreak.net Copyright (C) 2018 MyungJoo Ham myungjoo.ham@samsung.com

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Library General Public License as published by the Free Software Foundation; version 2.1 of the License.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Library General Public License for more details.

Date

10 Jul 2018

See also

https://github.com/nnstreamer/nnstreamer

Author

MyungJoo Ham myungjoo.ham@samsung.com

Bug No known bugs.

9.69.2 Macro Definition Documentation

9.69.2.1 GST_IS_TENSOR_TRANSFORM

```
#define GST_IS_TENSOR_TRANSFORM(
              obj ) (G_TYPE_CHECK_INSTANCE_TYPE((obj),GST_TYPE_TENSOR_TRANSFORM))
```

Definition at line 48 of file gsttensor_transform.h.

9.69.2.2 GST_IS_TENSOR_TRANSFORM_CLASS

```
#define GST_IS_TENSOR_TRANSFORM_CLASS(
              klass ) (G_TYPE_CHECK_CLASS_TYPE((klass),GST_TYPE_TENSOR_TRANSFORM))
```

Definition at line 50 of file gsttensor_transform.h.

9.69.2.3 GST_TENSOR_TRANSFORM

```
#define GST_TENSOR_TRANSFORM(
              obj ) (G_TYPE_CHECK_INSTANCE_CAST((obj),GST_TYPE_TENSOR_TRANSFORM,GstTensorTransform))
```

Definition at line 44 of file gsttensor_transform.h.

9.69.2.4 GST_TENSOR_TRANSFORM_CAST

```
#define GST_TENSOR_TRANSFORM_CAST(
              obj ) ((GstTensorTransform *)(obj))
```

Definition at line 52 of file gsttensor_transform.h.

9.69.2.5 GST_TENSOR_TRANSFORM_CLASS

```
#define GST_TENSOR_TRANSFORM_CLASS(
              klass ) (G_TYPE_CHECK_CLASS_CAST((klass),GST_TYPE_TENSOR_TRANSFORM,GstTensorTransformClass))
```

Definition at line 46 of file gsttensor_transform.h.

9.69.2.6 GST_TYPE_TENSOR_TRANSFORM

```
#define GST_TYPE_TENSOR_TRANSFORM (gst_tensor_transform_get_type())
```

Definition at line 42 of file gsttensor_transform.h.

9.69.3 Typedef Documentation

9.69.3.1 GstTensorTransform

```
typedef struct _GstTensorTransform GstTensorTransform
```

Definition at line 54 of file gsttensor_transform.h.

9.69.3.2 GstTensorTransformClass

```
typedef struct _GstTensorTransformClass GstTensorTransformClass
```

Definition at line 55 of file gsttensor_transform.h.

9.69.3.3 tensor_transform_arithmetic

```
typedef struct _tensor_transform_arithmetic tensor_transform_arithmetic
```

Internal data structure for arithmetic mode.

9.69.3.4 tensor_transform_clamp

```
typedef struct _tensor_transform_clamp tensor_transform_clamp
```

Internal data structure for clamp mode.

9.69.3.5 tensor_transform_dimchg

```
typedef struct _tensor_transform_dimchg tensor_transform_dimchg
```

Internal data structure for dimchg mode.

9.69.3.6 tensor_transform_mode

```
typedef enum _tensor_transform_mode tensor_transform_mode
```

9.69.3.7 tensor_transform_padding

```
typedef struct _tensor_transform_padding tensor_transform_padding
```

Internal data structure for padding mode.

9.69.3.8 tensor_transform_stand

```
typedef struct _tensor_transform_stand tensor_transform_stand
```

Internal data structure for stand mode.

9.69.3.9 tensor_transform_transpose

```
typedef struct _tensor_transform_transpose tensor_transform_transpose
```

Internal data structure for transpose mode.

9.69.3.10 tensor_transform_typecast

```
typedef struct _tensor_transform_typecast tensor_transform_typecast
```

Internal data structure for typecast mode.

9.69.4 Enumeration Type Documentation

9.69.4.1 _tensor_transform_mode

```
enum _tensor_transform_mode
```

Enumerator

GTT_DIMCHG	
GTT_TYPECAST	
GTT_ARITHMETIC	
GTT_TRANSPOSE	
GTT_STAND	
GTT_CLAMP	
GTT_PADDING	
GTT_UNKNOWN	

Definition at line 57 of file gsttensor_transform.h.

9.69.4.2 tensor_transform_operator

```
enum tensor_transform_operator
```

Enumerator

GTT_OP_TYPECAST	
GTT_OP_ADD	
GTT_OP_MUL	
GTT_OP_DIV	
GTT_OP_UNKNOWN	

Definition at line 70 of file gsttensor_transform.h.

9.69.4.3 tensor_transform_padding_axis

```
enum tensor_transform_padding_axis
```

Enumerator

PADDING_LEFT	
PADDING_RIGHT	
PADDING_TOP	
PADDING_BOTTOM	
PADDING_FRONT	
PADDING_BACK	
PADDING_END	

Definition at line 87 of file gsttensor_transform.h.

9.69.4.4 tensor_transform_stand_mode

```
enum tensor_transform_stand_mode
```

Enumerator

STAND_DEFAULT	
STAND_DC_AVERAGE	
STAND_END	

Definition at line 80 of file gsttensor_transform.h.

9.69.5 Function Documentation

9.69.5.1 gst_tensor_transform_get_type()

```
GType gst_tensor_transform_get_type (
            void  )
```

Get Type function required for gst elements.

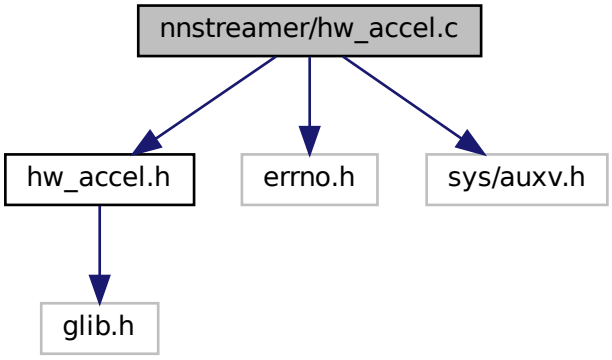
9.70 nnstreamer/elements/ignore_warning.sh File Reference

9.71 nnstreamer/hw_accel.c File Reference

Common hardware acceleration availability checks.

```
#include <hw_accel.h>
#include <errno.h>
#include <sys/auxv.h>
```

Include dependency graph for hw_accel.c:



Functions

- gint cpu_neon_accel_available (void)

  Check if neon is supported.

9.71.1 Detailed Description

Common hardware acceleration availability checks.

NNStreamer hardware accelerator availability checking Copyright (C) 2020 Parichay Kapoor pk.↩ kapoor@samsung.com

Date

8 September 2020

See also

https://github.com/nnstreamer/nnstreamer

Author

Parichay Kapoor pk.kapoor@samsung.com

Bug No known bugs except for NYI items

9.71.2 Function Documentation

9.71.2.1 cpu_neon_accel_available()

```
gint cpu_neon_accel_available (
            void  )
```

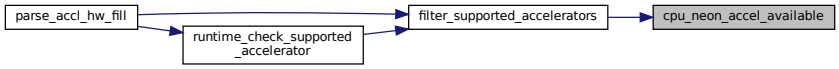
Check if neon is supported.

Return values

0	if supported, else -errno

Definition at line 41 of file hw_accel.c.

Here is the caller graph for this function:

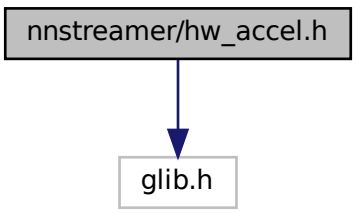


9.72 nnstreamer/hw_accel.h File Reference

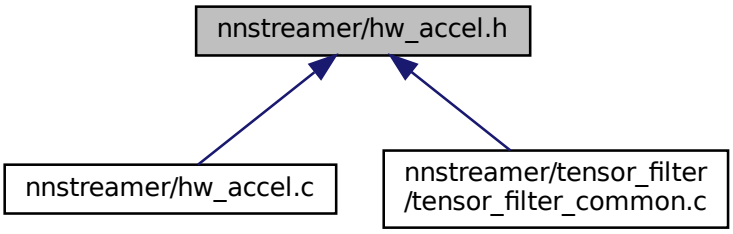
Common hardware acceleration availability header.

```
#include <glib.h>
```

Include dependency graph for hw_accel.h:



This graph shows which files directly or indirectly include this file:



Functions

- gint cpu_neon_accel_available (void)

  Check if neon is supported.

9.72.1 Detailed Description

Common hardware acceleration availability header.

NNStreamer hardware accelerator availability checking Copyright (C) 2020 Parichay Kapoor pk.↩ kapoor@samsung.com

Date

8 September 2020

See also

https://github.com/nnstreamer/nnstreamer

Author

Parichay Kapoor pk.kapoor@samsung.com

Bug No known bugs except for NYI items

9.72.2 Function Documentation

9.72.2.1 cpu_neon_accel_available()

```
gint cpu_neon_accel_available (
            void  )
```

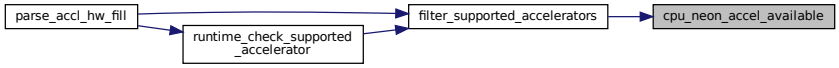
Check if neon is supported.

Return values

0	if supported, else -errno

Definition at line 41 of file hw_accel.c.

Here is the caller graph for this function:

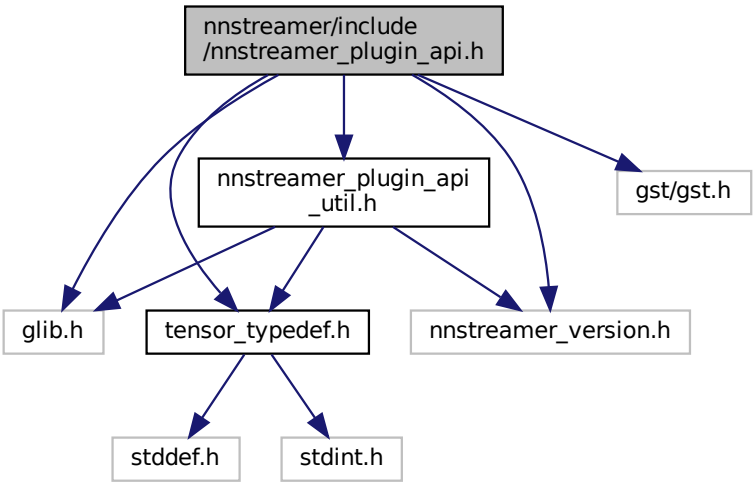


9.73 nnstreamer/include/nnstreamer_plugin_api.h File Reference

Optional/Additional NNStreamer APIs for sub-plugin writers. (Need Gst devel)

```
#include <glib.h>
#include <gst/gst.h>
#include <tensor_typedef.h>
#include <nnstreamer_version.h>
#include <nnstreamer_plugin_api_util.h>
```

Include dependency graph for nnstreamer_plugin_api.h:



This graph shows which files directly or indirectly include this file:

Functions

- G_BEGIN_DECLS gboolean gst_structure_is_tensor_stream (const GstStructure *structure)
  Check given mimetype is tensor stream.
- media_type gst_structure_get_media_type (const GstStructure *structure)
  Get media type from structure.
- gboolean gst_tensors_config_from_structure (GstTensorsConfig *config, const GstStructure *structure)
  Parse structure and set tensors config (for other/tensors)
- gboolean gst_tensors_config_from_peer (GstPad *pad, GstTensorsConfig *config, gboolean *is_fixed)
  Parse caps from peer pad and set tensors config.
- GstCaps * gst_tensor_caps_from_config (const GstTensorsConfig *config)
  Get tensor caps from tensors config (for other/tensor)
- GstCaps * gst_tensors_caps_from_config (const GstTensorsConfig *config)
  Get caps from tensors config (for other/tensors)
- void gst_tensor_alloc_init (gsize alignment)
  set alignment that default allocator would align to
- gboolean gst_tensor_meta_info_parse_memory (GstTensorMetaInfo *meta, GstMemory *mem)
  Parse memory and fill the tensor meta.
- GstMemory * gst_tensor_meta_info_append_header (GstTensorMetaInfo *meta, GstMemory *mem)
  Append header to memory.
- void gst_tensor_caps_update_dimension (GstCaps *caps, GstCaps *filter)
  Update caps dimension for negotiation.
- gboolean gst_tensor_caps_can_intersect (GstCaps *caps1, GstCaps *caps2)
  Try intersecting @caps1 and @caps2 for tensor stream.
- GstMemory * gst_tensor_buffer_get_nth_memory (GstBuffer *buffer, const guint index)
  Get the nth GstMemory from given buffer.
- gboolean gst_tensor_buffer_append_memory (GstBuffer *buffer, GstMemory *memory, const GstTensorInfo *info)
  Append memory to given buffer.
- guint gst_tensor_buffer_get_count (GstBuffer *buffer)
  Get the number of tensors in the buffer.

9.73.1 Detailed Description

Optional/Additional NNStreamer APIs for sub-plugin writers. (Need Gst devel)

NNStreamer Common API Header for Plug-Ins Copyright (C) 2019 MyungJoo Ham myungjoo.ham@samsung.com Copyright (C) 2019 Wook Song wook16.song@samsung.com

Date

24 Jan 2019

See also

https://github.com/nnstreamer/nnstreamer

Author

MyungJoo Ham myungjoo.ham@samsung.com and Wook Song wook16.song@samsung.com

Bug No known bugs except for NYI items

9.73.2 Function Documentation

9.73.2.1 gst_structure_get_media_type()

```
media_type gst_structure_get_media_type (
            const GstStructure * structure )
```

Get media type from structure.

Parameters

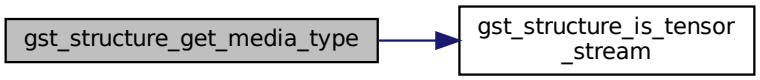
structure	structure to be interpreted

Returns

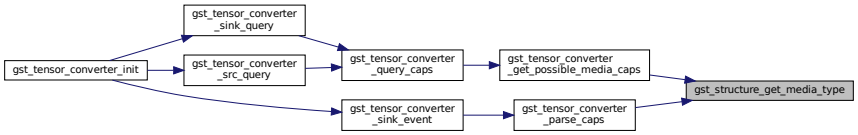
corresponding media type (returns _NNS_MEDIA_INVALID for unsupported type)

Definition at line 1001 of file nnstreamer_plugin_api_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.73.2.2 gst_structure_is_tensor_stream()

```
G_BEGIN_DECLS gboolean gst_structure_is_tensor_stream (
            const GstStructure * structure )
```

Check given mimetype is tensor stream.

Parameters

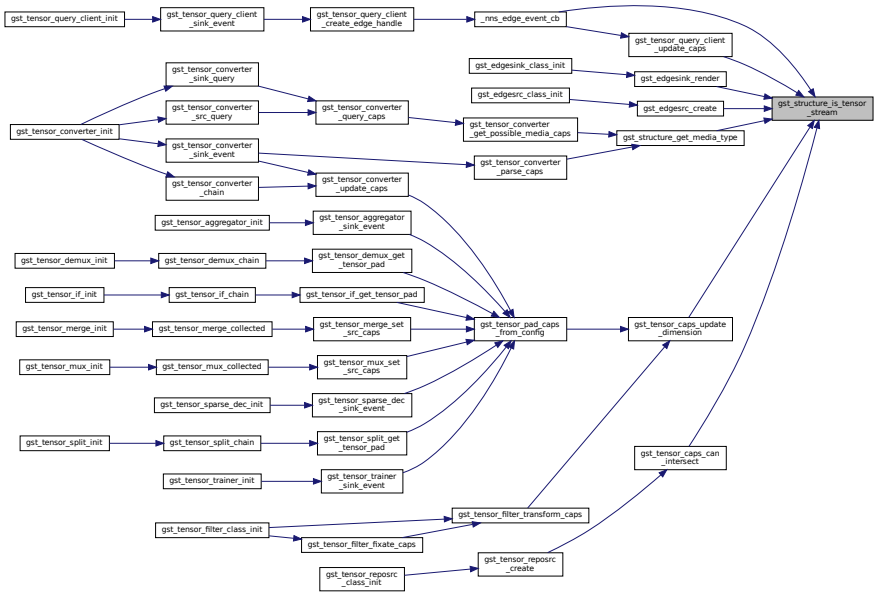
structure	structure to be interpreted

Returns

TRUE if mimetype is tensor stream

Definition at line 984 of file nnstreamer_plugin_api_impl.c.

Here is the caller graph for this function:



9.73.2.3 gst_tensor_alloc_init()

```
void gst_tensor_alloc_init (
            gsize alignment )
```

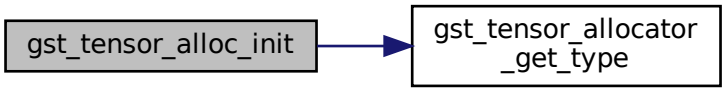
set alignment that default allocator would align to

Parameters

alignment	bytes of alignment

Definition at line 109 of file tensor_allocator.c.

Here is the call graph for this function:



9.73.2.4 gst_tensor_buffer_append_memory()

```
gboolean gst_tensor_buffer_append_memory (
            GstBuffer * buffer,
            GstMemory * memory,
            const GstTensorInfo * info )
```

Append *memory* to given *buffer*.

Parameters

	[in/out]	buffer GstBuffer to be appended.
in	*memory*	GstMemory to append. This function takes ownership of this, even if it returns failure.
in	*info*	GstTensorInfo of given *memory*.

Returns

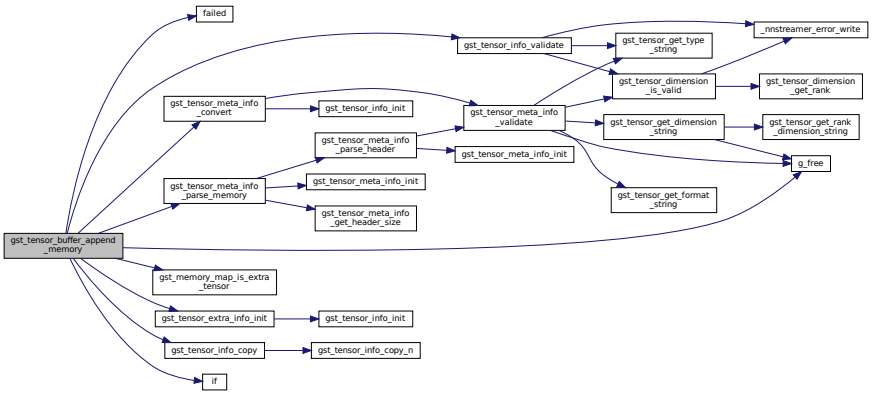
TRUE if successfully appended, otherwise FALSE.

Free the name string, cause it does not freed by gstreamer.

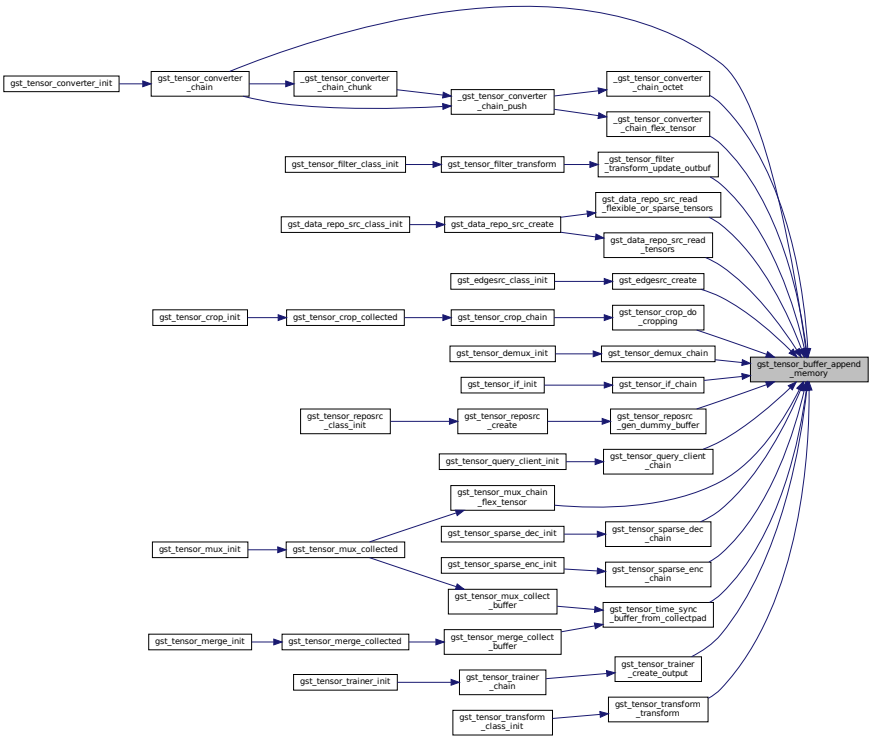
Todo Make custom gst_allocator later?

Definition at line 1666 of file nnstreamer_plugin_api_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



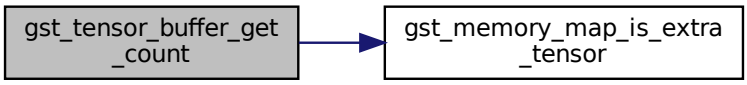
9.73.2.5 gst_tensor_buffer_get_count()

```
guint gst_tensor_buffer_get_count (
            GstBuffer * buffer )
```

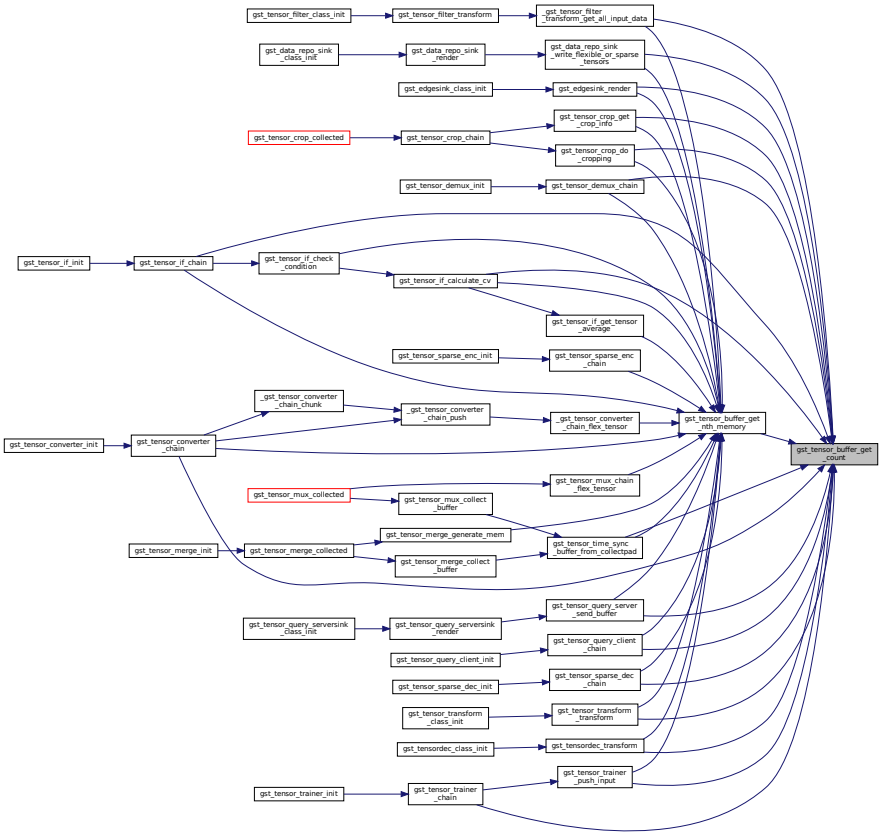
Get the number of tensors in the buffer.

Definition at line 1813 of file nnstreamer_plugin_api_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.73.2.6 gst_tensor_buffer_get_nth_memory()

```
GstMemory* gst_tensor_buffer_get_nth_memory (
            GstBuffer * buffer,
            const guint index )
```

Get the nth GstMemory from given *buffer*.

Parameters

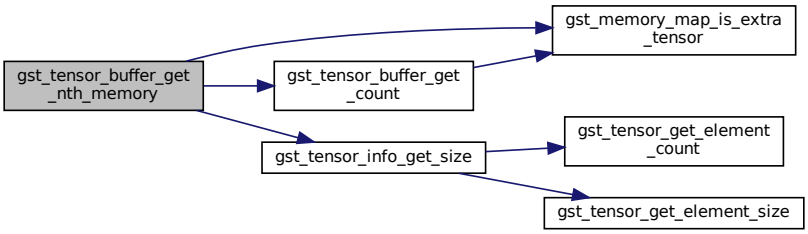
in	*buffer*	GstBuffer to be parsed.
in	*index*	Index of GstMemory to be returned.

Returns

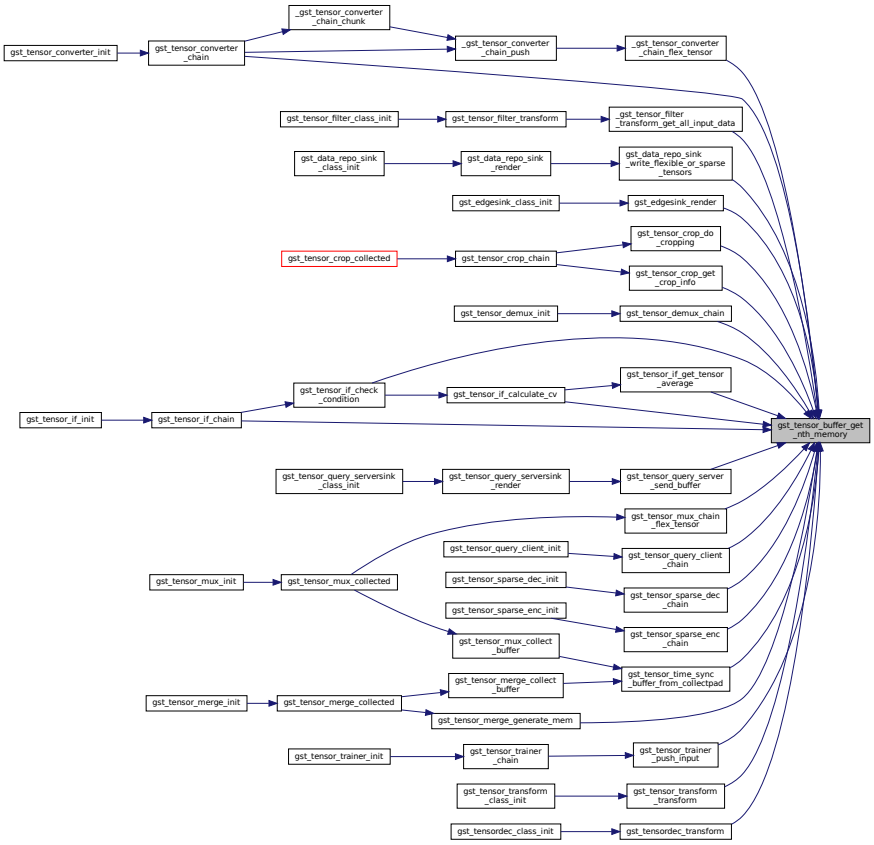
GstMemory if found, otherwise NULL (Caller should free returned memory using gst_memory_unref()).

Definition at line 1586 of file nnstreamer_plugin_api_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.73.2.7 gst_tensor_caps_can_intersect()

```
gboolean gst_tensor_caps_can_intersect (
            GstCaps * caps1,
            GstCaps * caps2 )
```

Try intersecting @caps1 and @caps2 for tensor stream.

Parameters

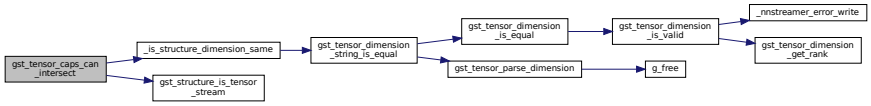
caps1	a GstCaps to intersect
caps2	a GstCaps to intersect

Returns

TRUE if intersection would be not empty.

Definition at line 1142 of file nnstreamer_plugin_api_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:

9.73.2.8 gst_tensor_caps_from_config()

```
GstCaps* gst_tensor_caps_from_config (
            const GstTensorsConfig * config )
```

Get tensor caps from tensors config (for other/tensor)

Parameters

config	tensors config info

Returns

caps for given config

Get tensor caps from tensors config (for other/tensor)

Parameters

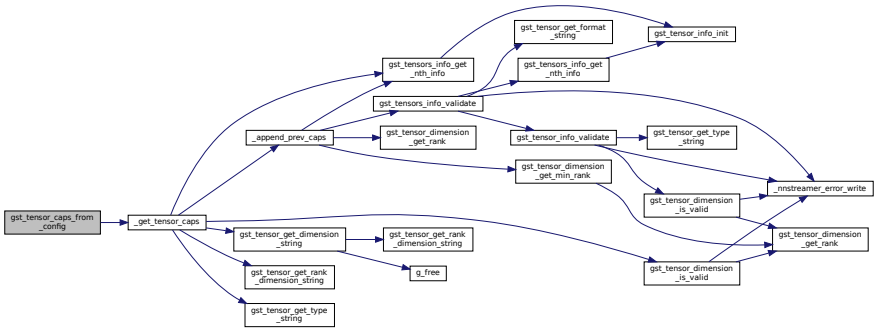
config	tensors config info

Returns

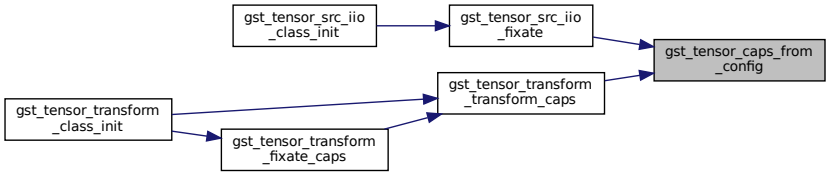
caps for given config

Definition at line 1395 of file nnstreamer_plugin_api_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.73.2.9 gst_tensor_caps_update_dimension()

```
void gst_tensor_caps_update_dimension (
            GstCaps * caps,
            GstCaps * filter )
```

Update caps dimension for negotiation.

Parameters

caps	caps to compare and update
filter	caps to compare

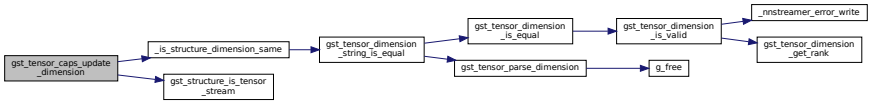
Update caps dimension for negotiation.

Parameters

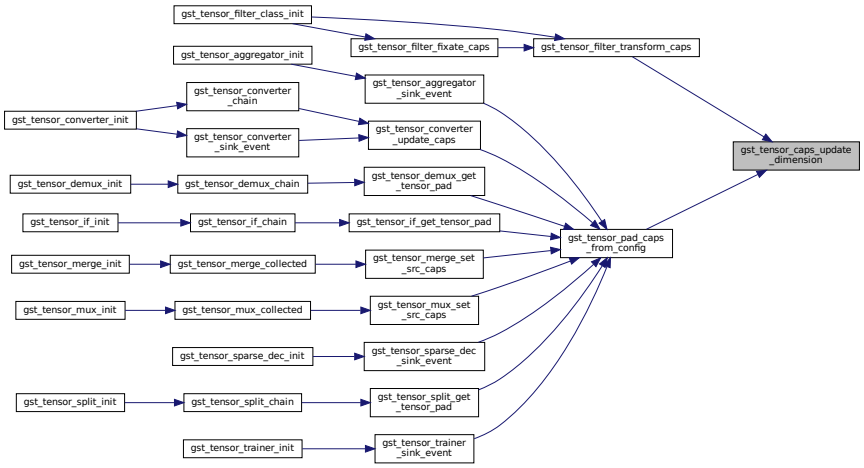
caps	caps to compare and update
filter	caps to compare

Definition at line 1093 of file nnstreamer_plugin_api_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.73.2.10 gst_tensor_meta_info_append_header()

```
GstMemory* gst_tensor_meta_info_append_header (
            GstTensorMetaInfo * meta,
            GstMemory * mem )
```

Append header to memory.

Parameters

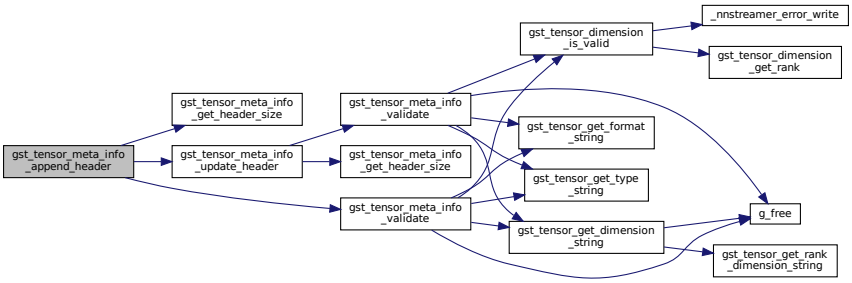
in	*meta*	tensor meta structure
in	*mem*	pointer to GstMemory

Returns

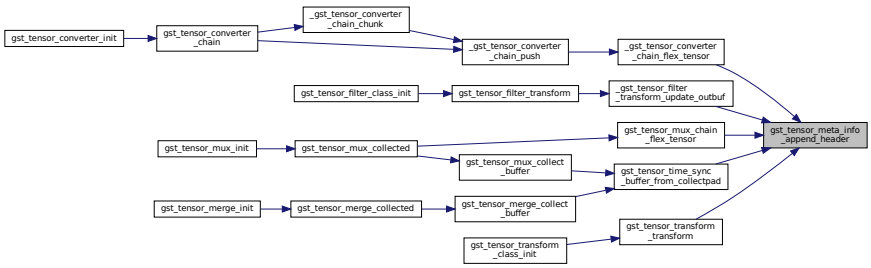
Newly allocated GstMemory (Caller should free returned memory using gst_memory_unref())

Definition at line 1544 of file nnstreamer_plugin_api_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.73.2.11 gst_tensor_meta_info_parse_memory()

```
gboolean gst_tensor_meta_info_parse_memory (
            GstTensorMetaInfo * meta,
            GstMemory * mem )
```

Parse memory and fill the tensor meta.

Parameters

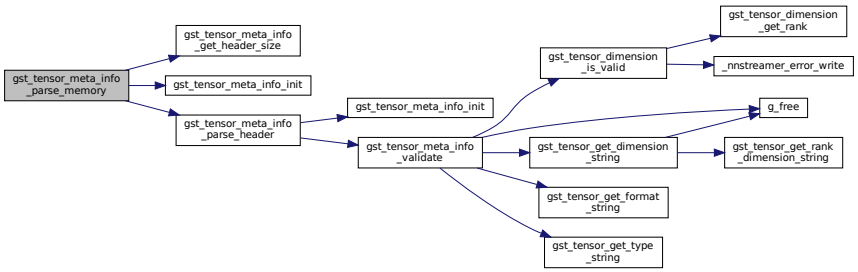
out	*meta*	tensor meta structure to be filled
in	*mem*	pointer to GstMemory to be parsed

Returns

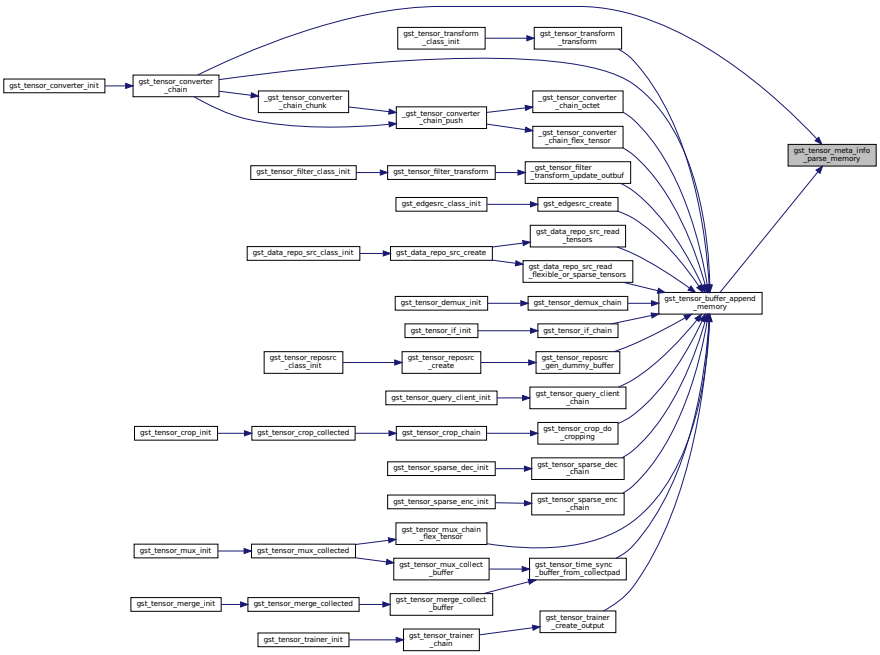
TRUE if successfully set the meta

Definition at line 1509 of file nnstreamer_plugin_api_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.73.2.12 gst_tensors_caps_from_config()

```
GstCaps* gst_tensors_caps_from_config (
            const GstTensorsConfig * config )
```

Get caps from tensors config (for other/tensors)

Parameters

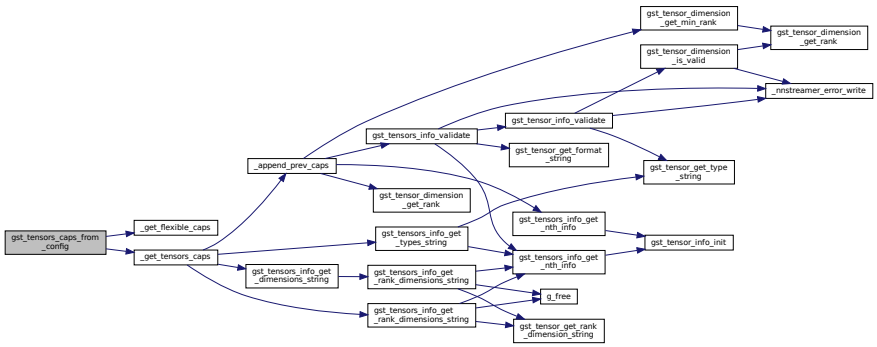
config	tensors config info

Returns

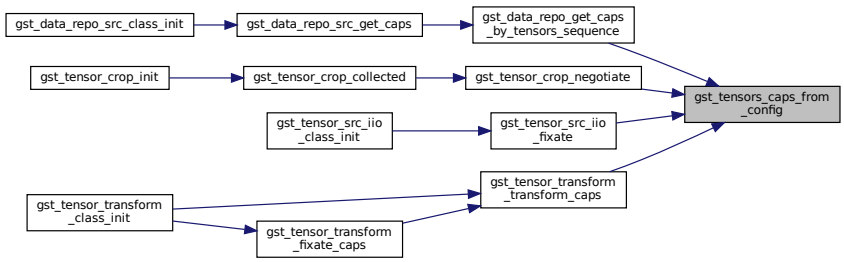
caps for given config

Definition at line 1372 of file nnstreamer_plugin_api_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.73.2.13 gst_tensors_config_from_peer()

```
gboolean gst_tensors_config_from_peer (
            GstPad * pad,
            GstTensorsConfig * config,
            gboolean * is_fixed )
```

Parse caps from peer pad and set tensors config.

Parameters

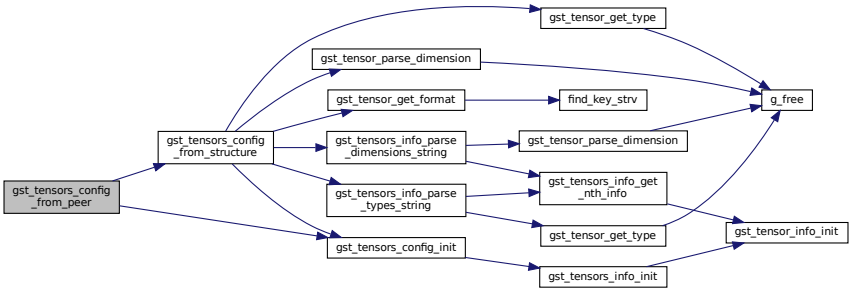
pad	GstPad to get the capabilities
config	tensors config structure to be filled
is_fixed	flag to be updated when peer caps is fixed (not mandatory, do nothing when the param is null)

Returns

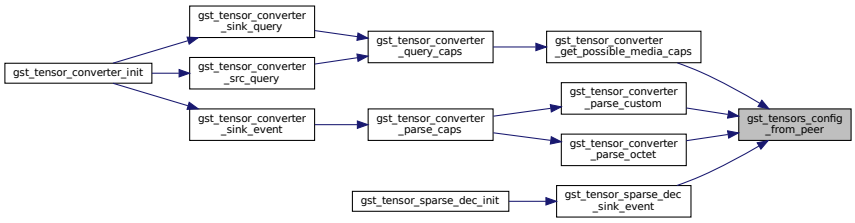
TRUE if successfully configured from peer

Definition at line 1041 of file nnstreamer_plugin_api_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.73.2.14 gst_tensors_config_from_structure()

```
gboolean gst_tensors_config_from_structure (
            GstTensorsConfig * config,
            const GstStructure * structure )
```

Parse structure and set tensors config (for other/tensors)

Parameters

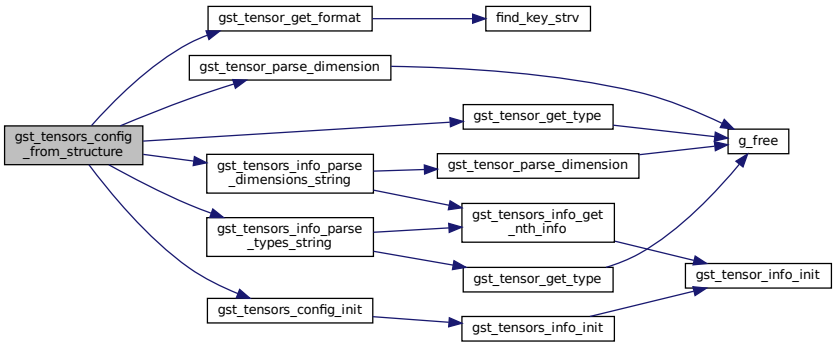
config	tensors config structure to be filled
structure	structure to be interpreted

Returns

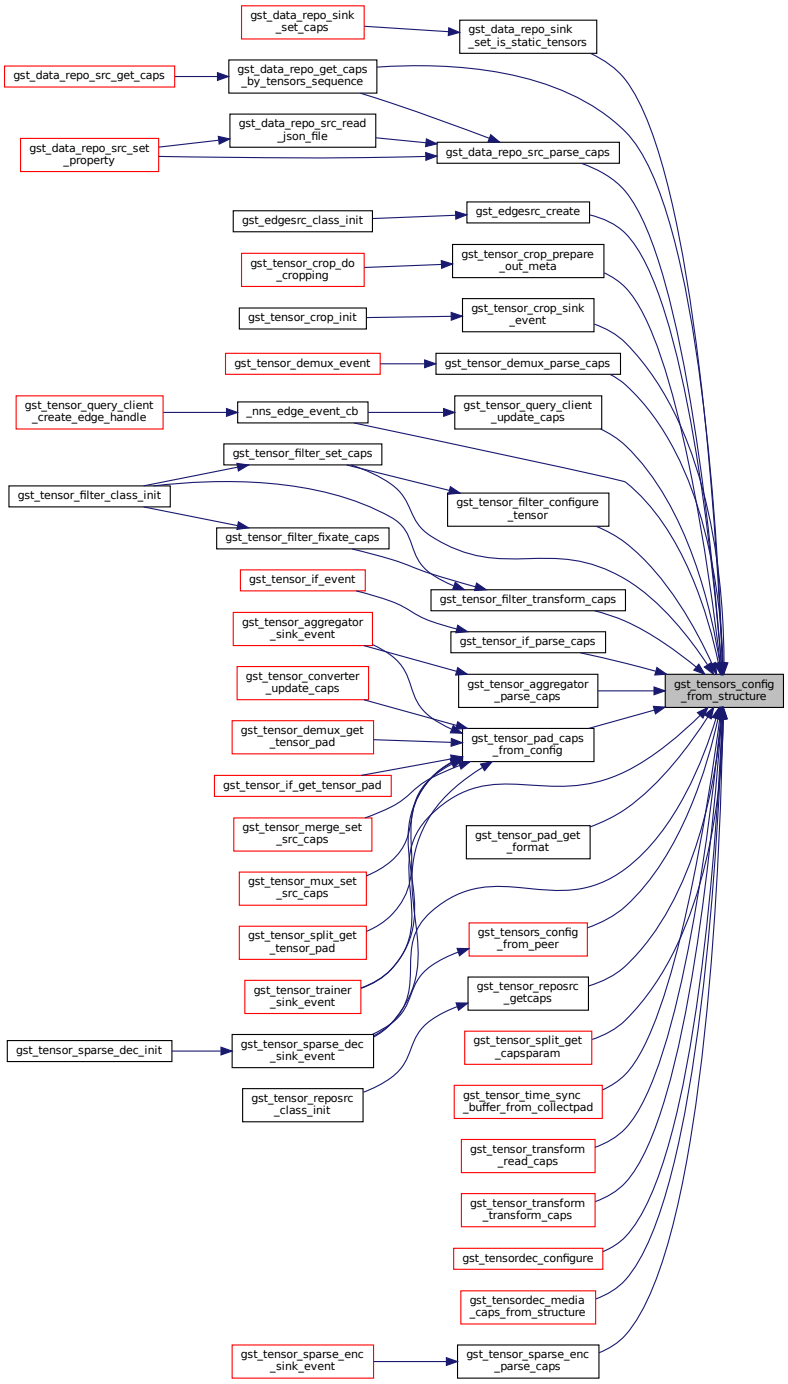
TRUE if no error

Definition at line 1413 of file nnstreamer_plugin_api_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:

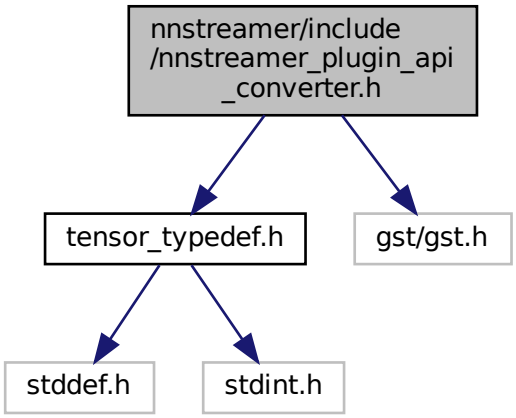


9.74 nnstreamer/include/nnstreamer_plugin_api_converter.h File Reference

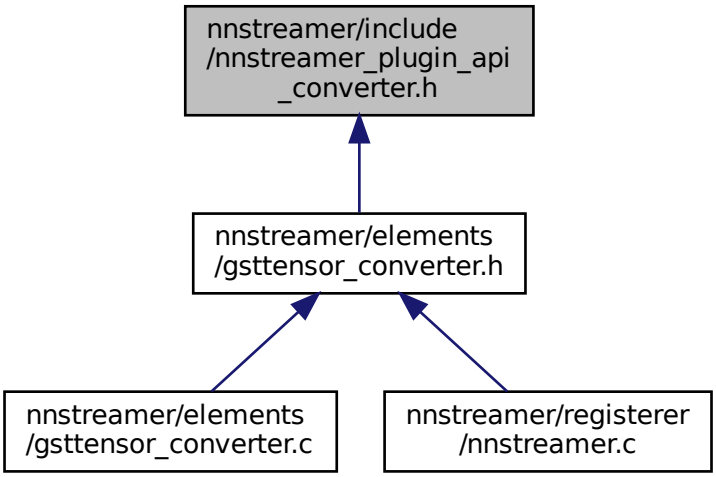
Mandatory APIs for NNStreamer Converter sub-plugins (Need Gst Devel)

```
#include "tensor_typedef.h"
#include <gst/gst.h>
```

Include dependency graph for nnstreamer_plugin_api_converter.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct _NNStreamerExternalConverter

  Converter's subplugin implementation.

Typedefs

- typedef struct _NNStreamerExternalConverter NNStreamerExternalConverter

  Converter's subplugin implementation.

Functions

- const NNStreamerExternalConverter * nnstreamer_converter_find (const char *name)

  Find converter sub-plugin with the name.
- int registerExternalConverter (NNStreamerExternalConverter *ex)

  Converter's sub-plugin should call this function to register itself.
- void unregisterExternalConverter (const char *prefix)

  Converter's sub-plugin may call this to unregister itself.
- void nnstreamer_converter_set_custom_property_desc (const char *name, const char *prop,...)

  set custom property description for tensor converter sub-plugin

9.74.1 Detailed Description

Mandatory APIs for NNStreamer Converter sub-plugins (Need Gst Devel)

NNStreamer API for Tensor_Converter Sub-Plugins Copyright (C) 2019 MyungJoo Ham myungjoo.ham@samsung.com

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Library General Public License as published by the Free Software Foundation; version 2.1 of the License.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Library General Public License for more details.

Date

09 Dec 2019

See also

https://github.com/nnstreamer/nnstreamer

Author

MyungJoo Ham myungjoo.ham@samsung.com

Bug No known bugs except for NYI items

9.74.2 Typedef Documentation

9.74.2.1 NNStreamerExternalConverter

```
typedef struct _NNStreamerExternalConverter NNStreamerExternalConverter
```

Converter's subplugin implementation.

9.74.3 Function Documentation

9.74.3.1 nnstreamer_converter_find()

```
const NNStreamerExternalConverter* nnstreamer_converter_find (
            const char * name )
```

Find converter sub-plugin with the name.

Parameters

in	*name*	The name of converter sub-plugin.

Returns

NNStreamerExternalConverter if subplugin is found. NULL if not found or the sub-plugin object has an error.

Parameters

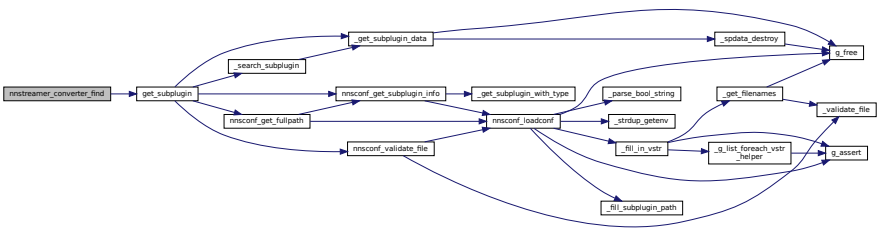
in	*name*	The name of converter sub-plugin.

Returns

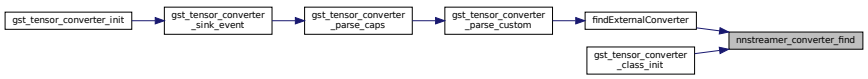
NULL if not found or the sub-plugin object has an error.

Definition at line 2360 of file gsttensor_converter.c.

Here is the call graph for this function:



Here is the caller graph for this function:



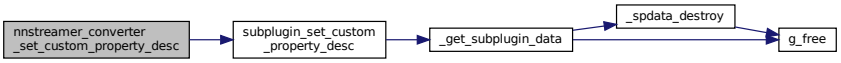
9.74.3.2 nnstreamer_converter_set_custom_property_desc()

```
void nnstreamer_converter_set_custom_property_desc (
            const char * name,
            const char * prop,
             ...  )
```

set custom property description for tensor converter sub-plugin

Definition at line 2457 of file gsttensor_converter.c.

Here is the call graph for this function:



9.74.3.3 registerExternalConverter()

```
int registerExternalConverter (
            NNStreamerExternalConverter * ex )
```

Converter's sub-plugin should call this function to register itself.

Parameters

in	*ex*	Converter sub-plugin to be registered.

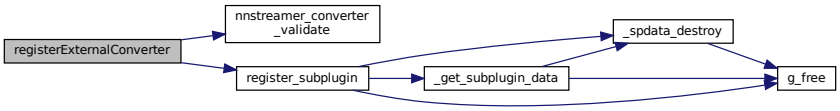
Returns

TRUE if registered. FALSE is failed or duplicated.

Converter's sub-plugin should call this function to register itself.

Definition at line 2389 of file gsttensor_converter.c.

Here is the call graph for this function:



9.74.3.4 unregisterExternalConverter()

```
void unregisterExternalConverter (
            const char * name )
```

Converter's sub-plugin may call this to unregister itself.

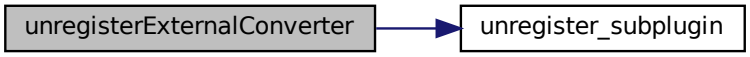
Parameters

in	*prefix*	The name of converter sub-plugin.

Converter's sub-plugin may call this to unregister itself.

Definition at line 2399 of file gsttensor_converter.c.

Here is the call graph for this function:

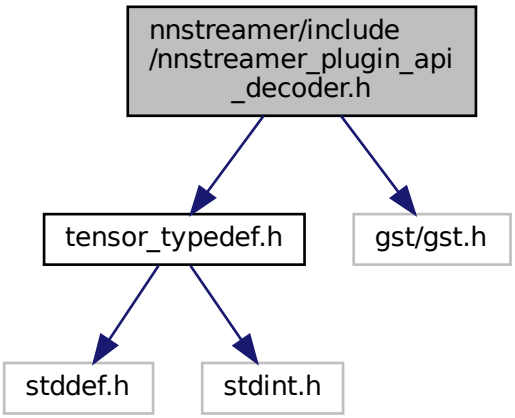


9.75 nnstreamer/include/nnstreamer_plugin_api_decoder.h File Reference

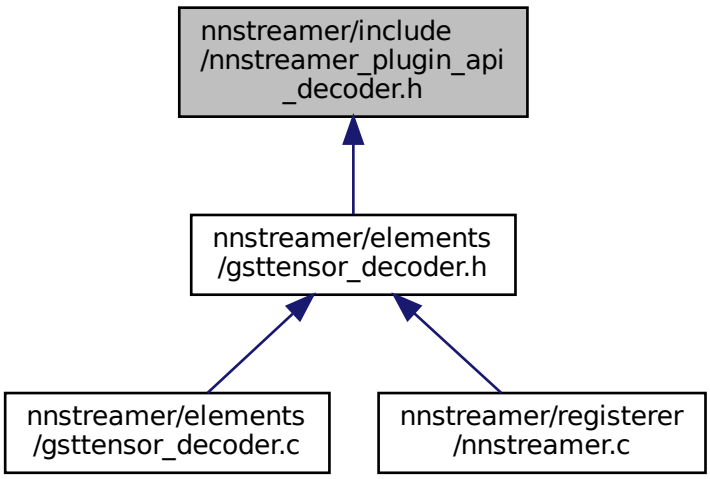
Mandatory APIs for NNStreamer Decoder sub-plugins (Need Gst Devel)

```
#include "tensor_typedef.h"
#include <gst/gst.h>
```

Include dependency graph for nnstreamer_plugin_api_decoder.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct _GstTensorDecoderDef

  Decoder definitions for different semantics of tensors This allows developers to create their own decoders.

Typedefs

- typedef struct _GstTensorDecoderDef GstTensorDecoderDef

  Decoder definitions for different semantics of tensors This allows developers to create their own decoders.

Functions

- int nnstreamer_decoder_probe (GstTensorDecoderDef *decoder)

  Decoder's sub-plugin should call this function to register itself.
- void nnstreamer_decoder_exit (const char *name)

  Decoder's sub-plugin may call this to unregister itself.
- const GstTensorDecoderDef * nnstreamer_decoder_find (const char *name)

  Find decoder sub-plugin with the name.
- void nnstreamer_decoder_set_custom_property_desc (const char *name, const char *prop,...)

  set custom property description for tensor decoder sub-plugin

9.75.1 Detailed Description

Mandatory APIs for NNStreamer Decoder sub-plugins (Need Gst Devel)

NNStreamer API for Tensor_Decoder Sub-Plugins Copyright (C) 2019 MyungJoo Ham myungjoo.ham@samsung.com

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Library General Public License as published by the Free Software Foundation; version 2.1 of the License.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Library General Public License for more details.

Date

30 Jan 2019

See also

https://github.com/nnstreamer/nnstreamer

Author

MyungJoo Ham myungjoo.ham@samsung.com

Bug No known bugs except for NYI items

9.75.2 Typedef Documentation

9.75.2.1 GstTensorDecoderDef

```
typedef struct _GstTensorDecoderDef GstTensorDecoderDef
```

Decoder definitions for different semantics of tensors This allows developers to create their own decoders.

9.75.3 Function Documentation

9.75.3.1 nnstreamer_decoder_exit()

```
void nnstreamer_decoder_exit (
            const char * name )
```

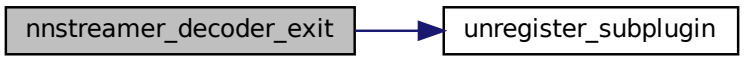
Decoder's sub-plugin may call this to unregister itself.

Parameters

in	*name*	The name of decoder sub-plugin.

Definition at line 166 of file gsttensor_decoder.c.

Here is the call graph for this function:



9.75.3.2 nnstreamer_decoder_find()

```
const GstTensorDecoderDef* nnstreamer_decoder_find (
            const char * name )
```

Find decoder sub-plugin with the name.

Parameters

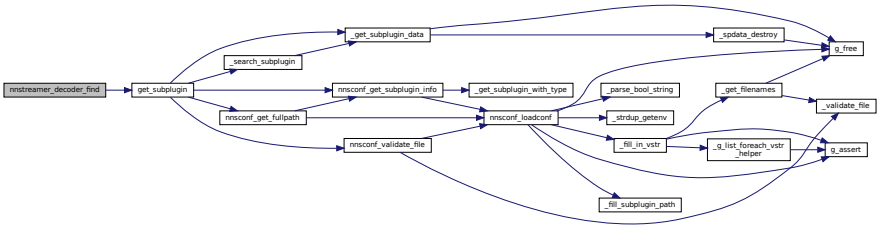
in	*name*	The name of decoder sub-plugin.

Returns

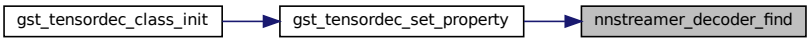
NULL if not found or the sub-plugin object has an error.

Definition at line 177 of file gsttensor_decoder.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.75.3.3 nnstreamer_decoder_probe()

```
int nnstreamer_decoder_probe (
            GstTensorDecoderDef * decoder )
```

Decoder's sub-plugin should call this function to register itself.

Parameters

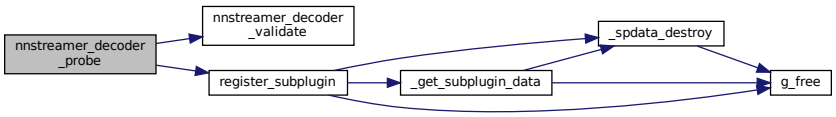
in	*decoder*	Decoder sub-plugin to be registered.

Returns

TRUE if registered. FALSE is failed or duplicated.

Definition at line 155 of file gsttensor_decoder.c.

Here is the call graph for this function:



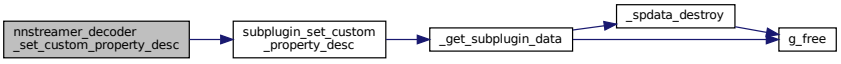
9.75.3.4 nnstreamer_decoder_set_custom_property_desc()

```
void nnstreamer_decoder_set_custom_property_desc (
            const char * name,
            const char * prop,
             ...  )
```

set custom property description for tensor decoder sub-plugin

Definition at line 186 of file gsttensor_decoder.c.

Here is the call graph for this function:

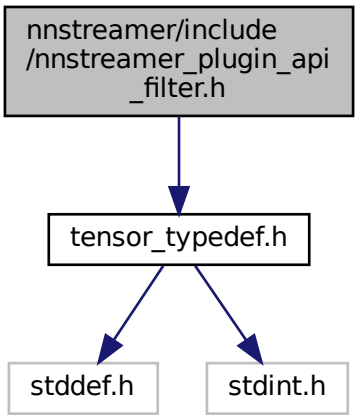


9.76 nnstreamer/include/nnstreamer_plugin_api_filter.h File Reference

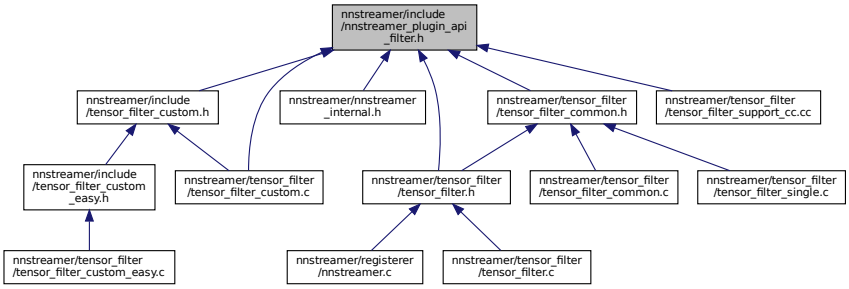
Mandatory APIs for NNStreamer Filter sub-plugins (No External Dependencies)

```
#include "tensor_typedef.h"
```

Include dependency graph for nnstreamer_plugin_api_filter.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct _GstTensorFilterProperties

  GstTensorFilter's properties for NN framework (internal data structure)
- struct _GstTensorFilterFrameworkStatistics

  Structure definition for tensor-filter framework statistics.
- struct _GstTensorFilterFrameworkInfo

  Tensor_Filter Subplugin framework related information.
- struct _GstTensorFilterFrameworkEventData

  User data for the tensor_tilter subplugin related events.
- struct _GstTensorFilterFramework

  Tensor_Filter Subplugin definition.
- struct parse_accl_args

  Accelerator related arguments for parsing.

Macros

- #define ACCL_NONE_STR "none"
- #define ACCL_DEFAULT_STR "default"
- #define ACCL_AUTO_STR "auto"
- #define ACCL_CPU_STR "cpu"
- #define ACCL_CPU_SIMD_STR "cpu.simd"
- #define ACCL_CPU_NEON_STR "cpu.neon"
- #define ACCL_GPU_STR "gpu"
- #define ACCL_NPU_STR "npu"
- #define ACCL_NPU_MOVIDIUS_STR "npu.movidius"
- #define ACCL_NPU_EDGE_TPU_STR "npu.edgetpu"
- #define ACCL_NPU_VIVANTE_STR "npu.vivante"
- #define ACCL_NPU_SRCN_STR "npu.srcn" /** srcn hardware supported by nnfw */
- #define ACCL_NPU_SLSI_STR "npu.slsi"
- #define ACCL_NPU_SR_STR "npu.sr"
- #define GST_TENSOR_FILTER_FRAMEWORK_BASE (0xDEAFDEAD00000000ULL)
- #define GST_TENSOR_FILTER_FRAMEWORK_V0 (GST_TENSOR_FILTER_FRAMEWORK_BASE)
- #define GST_TENSOR_FILTER_FRAMEWORK_V1 (GST_TENSOR_FILTER_FRAMEWORK_BASE | 0x10000ULL)
- #define GST_TENSOR_FILTER_API_VERSION_DEFINED (1)
- #define GST_TENSOR_FILTER_API_VERSION_MIN (0) /* The minimum API version supported (could be obsolete) */
- #define GST_TENSOR_FILTER_API_VERSION_MAX (1) /* The maximum API version supported (recommended) */
- #define checkGstTensorFilterFrameworkVersion(value, version) ((GST_TENSOR_FILTER_FRAMEWORK_BASE | ((version) << 16)) == (value & 0xFFFFFFFFFFFF0000ULL))

  Check the value of the version field of GstTensorFilterFramework.
- #define parse_accl_hw(...) parse_accl_hw_fill((parse_accl_args){__VA_ARGS__})

  workaround to provide default arguments

Typedefs

- typedef tensor_layout tensors_layout[NNS_TENSOR_SIZE_LIMIT]
- typedef struct _GstTensorFilterProperties GstTensorFilterProperties

  GstTensorFilter's properties for NN framework (internal data structure)
- typedef struct _GstTensorFilterFrameworkStatistics GstTensorFilterFrameworkStatistics

  Structure definition for tensor-filter framework statistics.
- typedef struct _GstTensorFilterFrameworkInfo GstTensorFilterFrameworkInfo

  Tensor_Filter Subplugin framework related information.
- typedef struct _GstTensorFilterFrameworkEventData GstTensorFilterFrameworkEventData

  User data for the tensor_tilter subplugin related events.
- typedef struct _GstTensorFilterFramework GstTensorFilterFramework

Enumerations

- enum accl_hw {
  ACCL_NONE = 0, ACCL_AUTO = 0x1, ACCL_DEFAULT = 0x2, ACCL_CPU = 0x1000,
  ACCL_CPU_SIMD = 0x1100, ACCL_CPU_NEON = 0x1100, ACCL_GPU = 0x2000, ACCL_NPU = 0x4000,
  ACCL_NPU_MOVIDIUS = 0x4001, ACCL_NPU_EDGE_TPU = 0x4002, ACCL_NPU_VIVANTE = 0x4003,
  ACCL_NPU_SRCN = 0x4004,
  ACCL_NPU_SLSI = 0x4005, ACCL_NPU_SR = 0x4100 }

  acceleration hw properties.
- enum event_ops {
  DESTROY_NOTIFY, RELOAD_MODEL, CUSTOM_PROP, SET_INPUT_PROP,
  SET_OUTPUT_PROP, SET_ACCELERATOR, CHECK_HW_AVAILABILITY }

  Tensor_Filter Subplugin related events.
- enum model_info_ops { GET_IN_OUT_INFO, SET_INPUT_INFO }

  Tensor_Filter Subplugin's model related info gathering operations.

Functions

- int nnstreamer_filter_probe (GstTensorFilterFramework *tfsp)

  Filter's sub-plugin should call this function to register itself.
- void nnstreamer_filter_exit (const char *name)

  Filter's sub-plugin may call this to unregister itself.
- const GstTensorFilterFramework * nnstreamer_filter_find (const char *name)

  Find filter sub-plugin with the name.
- void nnstreamer_filter_set_custom_property_desc (const char *name, const char *prop,...)

  set custom property description for tensor filter sub-plugin
- accl_hw get_accl_hw_type (const char *str)

  return accl_hw type from string
- const char * get_accl_hw_str (const accl_hw key)

  return string based on accl_hw type
- accl_hw parse_accl_hw_fill (parse_accl_args accl_args)

  parse user given string to extract accelerator based on given regex filling in optional arguments
- void * nnstreamer_filter_shared_model_get (void *instance, const char *key)

  Get the shared model representation that is already shared and has the same key.
- void * nnstreamer_filter_shared_model_insert_and_get (void *instance, char *key, void *interpreter)

  Insert the new shared model representation and get the value.
- int nnstreamer_filter_shared_model_remove (void *instance, const char *key, void(*free_callback)(void *))

  Remove the instance registered at the referred list of shared model table. If referred list is empty, `free_callback` is executed.
- void nnstreamer_filter_shared_model_replace (void *instance, const char *key, void *new_interpreter, void(*replace_callback)(void *, void *), void(*free_callback)(void *))

  Helper to reload interpreter for instances that has shared key. `replace_callback` is called iterating instances in referred list.

9.76.1 Detailed Description

Mandatory APIs for NNStreamer Filter sub-plugins (No External Dependencies)

NNStreamer API for Tensor_Filter Sub-Plugins Copyright (C) 2019 MyungJoo Ham myungjoo.ham@samsung.com

Date

30 Jan 2019

See also

https://github.com/nnstreamer/nnstreamer

Author

MyungJoo Ham myungjoo.ham@samsung.com

Bug No known bugs except for NYI items

9.76.2 Macro Definition Documentation

9.76.2.1 ACCL_AUTO_STR

```
#define ACCL_AUTO_STR "auto"
```

Definition at line 22 of file nnstreamer_plugin_api_filter.h.

9.76.2.2 ACCL_CPU_NEON_STR

```
#define ACCL_CPU_NEON_STR "cpu.neon"
```

Definition at line 25 of file nnstreamer_plugin_api_filter.h.

9.76.2.3 ACCL_CPU_SIMD_STR

```
#define ACCL_CPU_SIMD_STR "cpu.simd"
```

Definition at line 24 of file nnstreamer_plugin_api_filter.h.

9.76.2.4 ACCL_CPU_STR

```
#define ACCL_CPU_STR "cpu"
```

Definition at line 23 of file nnstreamer_plugin_api_filter.h.

9.76.2.5 ACCL_DEFAULT_STR

```
#define ACCL_DEFAULT_STR "default"
```

Definition at line 21 of file nnstreamer_plugin_api_filter.h.

9.76.2.6 ACCL_GPU_STR

```
#define ACCL_GPU_STR "gpu"
```

Definition at line 26 of file nnstreamer_plugin_api_filter.h.

9.76.2.7 ACCL_NONE_STR

```
#define ACCL_NONE_STR "none"
```

Macros for accelerator types

Definition at line 20 of file nnstreamer_plugin_api_filter.h.

9.76.2.8 ACCL_NPU_EDGE_TPU_STR

```
#define ACCL_NPU_EDGE_TPU_STR "npu.edgetpu"
```

Definition at line 30 of file nnstreamer_plugin_api_filter.h.

9.76.2.9 ACCL_NPU_MOVIDIUS_STR

```
#define ACCL_NPU_MOVIDIUS_STR "npu.movidius"
```

Definition at line 29 of file nnstreamer_plugin_api_filter.h.

9.76.2.10 ACCL_NPU_SLSI_STR

```
#define ACCL_NPU_SLSI_STR "npu.slsi"
```

Definition at line 33 of file nnstreamer_plugin_api_filter.h.

9.76.2.11 ACCL_NPU_SR_STR

```
#define ACCL_NPU_SR_STR "npu.sr"
```

Definition at line 34 of file nnstreamer_plugin_api_filter.h.

9.76.2.12 ACCL_NPU_SRCN_STR

```
#define ACCL_NPU_SRCN_STR "npu.srcn" /** srcn hardware supported by nnfw */
```

Definition at line 32 of file nnstreamer_plugin_api_filter.h.

9.76.2.13 ACCL_NPU_STR

```
#define ACCL_NPU_STR "npu"
```

Todo Define ACCL_DSP_STR

Definition at line 28 of file nnstreamer_plugin_api_filter.h.

9.76.2.14 ACCL_NPU_VIVANTE_STR

```
#define ACCL_NPU_VIVANTE_STR "npu.vivante"
```

Definition at line 31 of file nnstreamer_plugin_api_filter.h.

9.76.2.15 checkGstTensorFilterFrameworkVersion

```
#define checkGstTensorFilterFrameworkVersion(
              value,
              version ) ((GST_TENSOR_FILTER_FRAMEWORK_BASE | ((version) << 16)) == (value &
0xFFFFFFFFFFFF0000ULL))
```

Check the value of the version field of GstTensorFilterFramework.

Definition at line 47 of file nnstreamer_plugin_api_filter.h.

9.76.2.16 GST_TENSOR_FILTER_API_VERSION_DEFINED

```
#define GST_TENSOR_FILTER_API_VERSION_DEFINED (1)
```

Definition at line 40 of file nnstreamer_plugin_api_filter.h.

9.76.2.17 GST_TENSOR_FILTER_API_VERSION_MAX

```
#define GST_TENSOR_FILTER_API_VERSION_MAX (1) /* The maximum API version supported (recommended)
*/
```

Definition at line 42 of file nnstreamer_plugin_api_filter.h.

9.76.2.18 GST_TENSOR_FILTER_API_VERSION_MIN

```
#define GST_TENSOR_FILTER_API_VERSION_MIN (0) /* The minimum API version supported (could be
obsolete) */
```

Definition at line 41 of file nnstreamer_plugin_api_filter.h.

9.76.2.19 GST_TENSOR_FILTER_FRAMEWORK_BASE

```
#define GST_TENSOR_FILTER_FRAMEWORK_BASE (0xDEAFDEAD00000000ULL)
```

Definition at line 36 of file nnstreamer_plugin_api_filter.h.

9.76.2.20 GST_TENSOR_FILTER_FRAMEWORK_V0

```
#define GST_TENSOR_FILTER_FRAMEWORK_V0 (GST_TENSOR_FILTER_FRAMEWORK_BASE)
```

Definition at line 37 of file nnstreamer_plugin_api_filter.h.

9.76.2.21 GST_TENSOR_FILTER_FRAMEWORK_V1

```
#define GST_TENSOR_FILTER_FRAMEWORK_V1 (GST_TENSOR_FILTER_FRAMEWORK_BASE | 0x10000ULL)
```

Definition at line 38 of file nnstreamer_plugin_api_filter.h.

9.76.2.22 parse_accl_hw

```
#define parse_accl_hw(
              ...  ) parse_accl_hw_fill((parse_accl_args){__VA_ARGS__})
```

workaround to provide default arguments

Definition at line 544 of file nnstreamer_plugin_api_filter.h.

9.76.3 Typedef Documentation

9.76.3.1 GstTensorFilterFramework

```
typedef struct _GstTensorFilterFramework GstTensorFilterFramework
```

Definition at line 240 of file nnstreamer_plugin_api_filter.h.

9.76.3.2 GstTensorFilterFrameworkEventData

```
typedef struct _GstTensorFilterFrameworkEventData GstTensorFilterFrameworkEventData
```

User data for the tensor_tilter subplugin related events.

9.76.3.3 GstTensorFilterFrameworkInfo

```
typedef struct _GstTensorFilterFrameworkInfo GstTensorFilterFrameworkInfo
```

Tensor_Filter Subplugin framework related information.

All the information except the supported accelerator is provided statically. Accelerators can be provided based on static or dynamic check dependent on framework support.

9.76.3.4 GstTensorFilterFrameworkStatistics

```
typedef struct _GstTensorFilterFrameworkStatistics GstTensorFilterFrameworkStatistics
```

Structure definition for tensor-filter framework statistics.

9.76.3.5 GstTensorFilterProperties

```
typedef struct _GstTensorFilterProperties GstTensorFilterProperties
```

GstTensorFilter's properties for NN framework (internal data structure)

Because custom filters of tensor_filter may need to access internal data of GstTensorFilter, we define this data structure here.

9.76.3.6 tensors_layout

```
typedef tensor_layout tensors_layout[NNS_TENSOR_SIZE_LIMIT]
```

Definition at line 104 of file nnstreamer_plugin_api_filter.h.

9.76.4 Enumeration Type Documentation

9.76.4.1 accl_hw

```
enum accl_hw
```

acceleration hw properties.

Enabling acceleration (choosing any accelerator hardware other ACCL_NONE) will enable acceleration for the framework dependent on the acceleration supported by the framework.

For example, enabling acceleration with tflite will enable NNAPI. However, with NNFW will enable GPU/NEON etc.

Appropriate acceleration should be used with each framework. For example, ACCL_CPU_NEON is supported with NNFW tensor filter. Using ACCL_NEON with pytorch would result in a warning message, and the accelerator would fallback on ACCL_AUTO.

ACCL_AUTO automatically chooses the accelerator based on the ones supported by the subplugin framework. However, ACCL_DEFAULT would use the accelerator set by the subplugin framework, if any.

Note

Acceleration for a framework can be enabled/disabled in the build. If the acceleration for a framework was disabled in the build, setting the accelerator while use a tensor filter with that framework will have no effect.

Add definition of new accelerators to accl_hw_get_type() in tensor_filter_common.c as well.

Enumerator

ACCL_NONE	no explicit acceleration no acceleration (defaults to CPU)
ACCL_AUTO	If there is no default config, and device needs to be specified, fallback to ACCL_AUTO choose optimized device automatically
ACCL_DEFAULT	use default device configuration by the framework
ACCL_CPU	Enables acceleration, 0xn000 any version of that device, 0xnxxx: device # xxx-1 specify device as CPU, if possible

Enumerator

ACCL_CPU_SIMD	specify device as SIMD in cpu, if possible
ACCL_CPU_NEON	specify device as NEON (alias for SIMD) in cpu, if possible
ACCL_GPU	specify device as GPU, if possible
ACCL_NPU	**Todo** Define ACCL_DSP specify device as any NPU, if possible
ACCL_NPU_MOVIDIUS	specify device as movidius, if possible
ACCL_NPU_EDGE_TPU	specify device as edge tpu, if possible
ACCL_NPU_VIVANTE	specify device as vivante, if possible
ACCL_NPU_SRCN	specify device as srcn, if possible
ACCL_NPU_SLSI	specify device as S.LSI, if possible
ACCL_NPU_SR	specify device as any SR npu, if possible

Definition at line 80 of file nnstreamer_plugin_api_filter.h.

9.76.4.2 event_ops

`enum event_ops`

Tensor_Filter Subplugin related events.

These are possible set of events that can be supported by the tensor filter subplugin.

Enumerator

DESTROY_NOTIFY	Free the data element allocated in the invoke callback
RELOAD_MODEL	Reloads the subplugin with newly provided model
CUSTOM_PROP	Update the custom properties for the framework
SET_INPUT_PROP	Update input tensor info and layout
SET_OUTPUT_PROP	Update output tensor info and layout
SET_ACCELERATOR	Update accelerator of the subplugin to be used as backend
CHECK_HW_AVAILABILITY	Check the hw availability with custom option

Definition at line 177 of file nnstreamer_plugin_api_filter.h.

9.76.4.3 model_info_ops

`enum model_info_ops`

Tensor_Filter Subplugin's model related info gathering operations.

Enumerator

GET_IN_OUT_INFO	Gets the input and output tensor info
SET_INPUT_INFO	Sets the provided input tensor info, and get updated output tensor info

Definition at line 191 of file nnstreamer_plugin_api_filter.h.

9.76.5 Function Documentation

9.76.5.1 get_accl_hw_str()

```
const char* get_accl_hw_str (
            const accl_hw key )
```

return string based on accl_hw type

Parameters

key	The key enum value

Returns

Corresponding string. Returns ACCL_NONE_STR if not found.

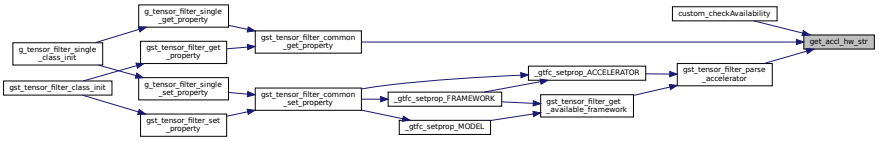
Note

Do not free the returned char *

Definition at line 2598 of file tensor_filter_common.c.

Here is the call graph for this function:

Here is the caller graph for this function:



9.76.5.2 get_accl_hw_type()

```
accl_hw get_accl_hw_type (
            const char * str )
```

return accl_hw type from string

9.76.5.3 nnstreamer_filter_exit()

```
void nnstreamer_filter_exit (
            const char * name )
```

Filter's sub-plugin may call this to unregister itself.

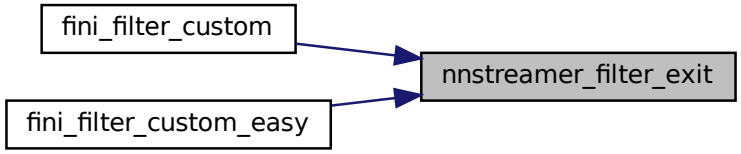
Parameters

in	*name*	The name of filter sub-plugin.

Definition at line 638 of file tensor_filter_common.c.

Here is the call graph for this function:

Here is the caller graph for this function:



9.76.5.4 nnstreamer_filter_find()

```
const GstTensorFilterFramework* nnstreamer_filter_find (
            const char * name )
```

Find filter sub-plugin with the name.

Parameters

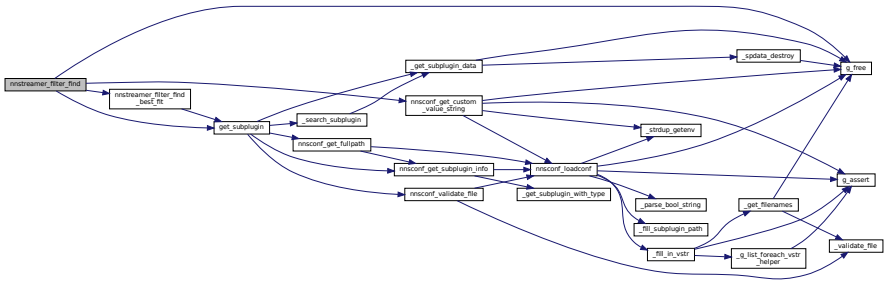
in	*name*	The name of filter sub-plugin.

Returns

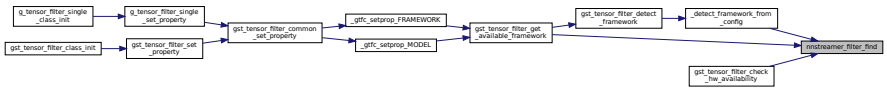
NULL if not found or the sub-plugin object has an error.

Definition at line 697 of file tensor_filter_common.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.76.5.5 nnstreamer_filter_probe()

```
int nnstreamer_filter_probe (
            GstTensorFilterFramework * tfsp )
```

Filter's sub-plugin should call this function to register itself.

Parameters

in	*tfsp*	Tensor-Filter Sub-Plugin to be registered.

Returns

TRUE if registered. FALSE is failed or duplicated.

Note

Do not change the subplugins callbacks after probing the filter.

Parameters

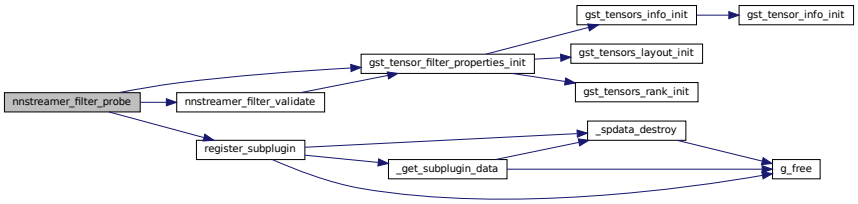
in	*tfsp*	Tensor-Filter Sub-Plugin to be registered.

Returns

TRUE if registered. FALSE is failed or duplicated.

Definition at line 611 of file tensor_filter_common.c.

Here is the call graph for this function:



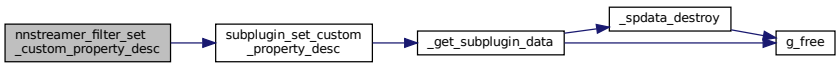
9.76.5.6 nnstreamer_filter_set_custom_property_desc()

```
void nnstreamer_filter_set_custom_property_desc (
            const char * name,
            const char * prop,
             ...  )
```

set custom property description for tensor filter sub-plugin

Definition at line 647 of file tensor_filter_common.c.

Here is the call graph for this function:



9.76.5.7 nnstreamer_filter_shared_model_get()

```
void* nnstreamer_filter_shared_model_get (
            void * instance,
            const char * key )
```

Get the shared model representation that is already shared and has the same key.

Parameters

in	*instance*	The instance that is sharing the model representation. It will be registered at the referred list.
in	*key*	The key to find the matched shared representation.

Returns

The model interpreter. NULL if it does not exist.

Definition at line 2993 of file tensor_filter_common.c.

9.76.5.8 nnstreamer_filter_shared_model_insert_and_get()

```
void* nnstreamer_filter_shared_model_insert_and_get (
            void * instance,
            char * key,
            void * interpreter )
```

Insert the new shared model representation and get the value.

Parameters

in	*instance*	The instance that is sharing the model representation. It will be registered at the referred list.
in	*key*	The key for shared model.
in	*interpreter*	The interpreter to be shared.

Returns

The model interpreter inserted. NULL if it is already inserted.

Internal error case. The interpreter already exists in shared table, do not insert and return null.

Definition at line 3026 of file tensor_filter_common.c.

9.76.5.9 nnstreamer_filter_shared_model_remove()

```
int nnstreamer_filter_shared_model_remove (
            void * instance,
```

```
            const char * key,
            void(*)(void *) free_callback )
```

Remove the instance registered at the referred list of shared model table. If referred list is empty, `free_callback` is executed.

Parameters

in	*instance*	The instance that should be removed from the referred list.
in	*key*	The key to find the shared model.
in	*free_callback*	The callback function to destroy the interpreter, which takes the interpreter as arg.

Returns

TRUE if the instance is removed. FALSE if failed to remove it.

Definition at line 3081 of file tensor_filter_common.c.

9.76.5.10 nnstreamer_filter_shared_model_replace()

```
void nnstreamer_filter_shared_model_replace (
            void * instance,
            const char * key,
            void * new_interpreter,
            void(*)(void *, void *) replace_callback,
            void(*)(void *) free_callback )
```

Helper to reload interpreter for instances that has shared key. `replace_callback` is called iterating instances in referred list.

Parameters

in	*instance*	The instance that is sharing the model representation.
in	*key*	The key to find the shared model.
in	*interpreter*	The new interpreter to replace.
in	*replace_callback*	The callback function to replace with new interpreter.
in	*free_callback*	The callback function to destroy the old interpreter.

Definition at line 3128 of file tensor_filter_common.c.

9.76.5.11 parse_accl_hw_fill()

```
accl_hw parse_accl_hw_fill (
            parse_accl_args accl_args )
```

parse user given string to extract accelerator based on given regex filling in optional arguments

Note

The order of arguments for calling this function is:

- in_accl: user provided input accelerator string
- sup_accl: list of supported accelerator
- auto_accl: auto accelerator (optional)
- def_accl: default accelerator (optional)

remove unsupported accelerators from this list based on runtime system

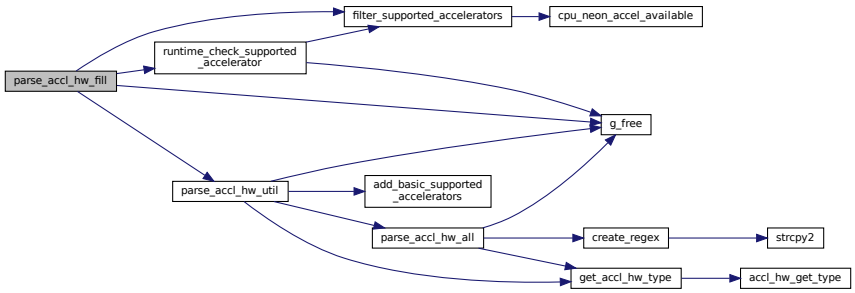
filtered supported accelerators can be empty

update default accelerator if it is not available at runtime

update auto accelerator if it is not available at runtime

Definition at line 2833 of file tensor_filter_common.c.

Here is the call graph for this function:

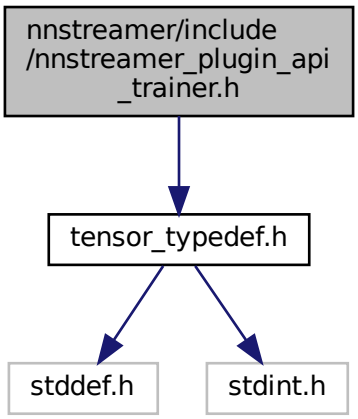


9.77 nnstreamer/include/nnstreamer_plugin_api_trainer.h File Reference

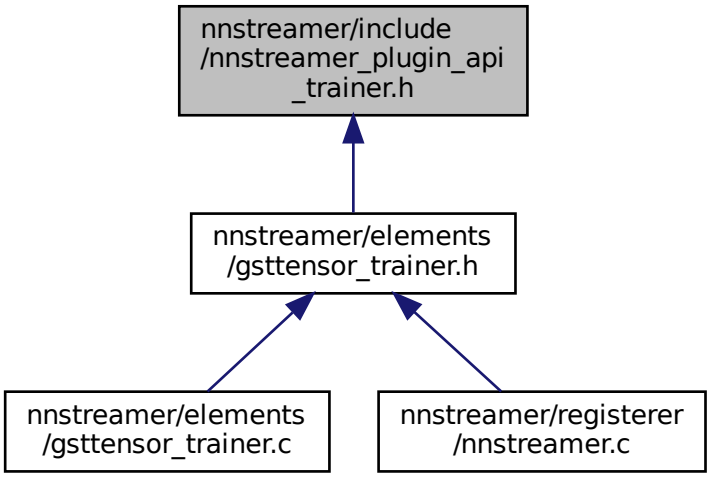
Mandatory APIs for NNStreamer Trainer sub-plugins (No External Dependencies)

```
#include "tensor_typedef.h"
```

Include dependency graph for nnstreamer_plugin_api_trainer.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct _GstTensorTrainerProperties

  GstTensorTrainer's properties for neural network framework (internal data structure)
- struct _GstTensorTrainerFrameworkInfo

  GstTensorTrainer's subplugin framework related information.
- struct _GstTensorTrainerEventNotifier

  GstTensorTrainer's event notifier.
- struct _GstTensorTrainerFramework

  tensor_trainer subplugin definition

Macros

- #define GST_TENSOR_TRAINER_FRAMEWORK_BASE (0xDEAFDEAD00000000ULL)
- #define GST_TENSOR_TRAINER_FRAMEWORK_V1 (GST_TENSOR_TRAINER_FRAMEWORK_BASE | 0x10000ULL)

Typedefs

- typedef struct _GstTensorTrainerProperties GstTensorTrainerProperties

  GstTensorTrainer's properties for neural network framework (internal data structure)
- typedef struct _GstTensorTrainerFrameworkInfo GstTensorTrainerFrameworkInfo

  GstTensorTrainer's subplugin framework related information.
- typedef struct _GstTensorTrainerFramework GstTensorTrainerFramework
- typedef struct _GstTensorTrainerEventNotifier GstTensorTrainerEventNotifier

  GstTensorTrainer's event notifier.

Enumerations

- enum GstTensorTrainerEventType { TRAINER_EVENT_EPOCH_COMPLETION, TRAINER_EVENT_TRAINING_COMPLETIO TRAINER_EVENT_UNKNOWN }

  GstTensorTrainer's event type list.

Functions

- int nnstreamer_trainer_probe (GstTensorTrainerFramework *ttsp)

  Trainer's sub-plugin should call this function to register itself.
- int nnstreamer_trainer_exit (GstTensorTrainerFramework *ttsp)

  Trainer's sub-plugin may call this to unregister itself.
- void nnstreamer_trainer_notify_event (GstTensorTrainerEventNotifier *notifier, GstTensorTrainerEventType type, void *data)

  Trainer's sub-plugin call this to notify event.

9.77.1 Detailed Description

Mandatory APIs for NNStreamer Trainer sub-plugins (No External Dependencies)

NNStreamer API for Tensor_Trainer Sub-Plugins Copyright (C) 2022 Hyunil Park hyunil46.park@samsung.↩ com

Date

1 Dec 2022

See also

https://github.com/nnstreamer/nnstreamer

Author

Hyunil Park hyunil46.park@samsung.com

Bug No known bugs except for NYI items

9.77.2 Macro Definition Documentation

9.77.2.1 GST_TENSOR_TRAINER_FRAMEWORK_BASE

```
#define GST_TENSOR_TRAINER_FRAMEWORK_BASE (0xDEAFDEAD00000000ULL)
```

Definition at line 19 of file nnstreamer_plugin_api_trainer.h.

9.77.2.2 GST_TENSOR_TRAINER_FRAMEWORK_V1

```
#define GST_TENSOR_TRAINER_FRAMEWORK_V1 (GST_TENSOR_TRAINER_FRAMEWORK_BASE | 0x10000ULL)
```

Definition at line 20 of file nnstreamer_plugin_api_trainer.h.

9.77.3 Typedef Documentation

9.77.3.1 GstTensorTrainerEventNotifier

```
typedef struct _GstTensorTrainerEventNotifier GstTensorTrainerEventNotifier
```

GstTensorTrainer's event notifier.

Subplugins must send events to tensor_trainer with a notifier.

9.77.3.2 GstTensorTrainerFramework

```
typedef struct _GstTensorTrainerFramework GstTensorTrainerFramework
```

Definition at line 60 of file nnstreamer_plugin_api_trainer.h.

9.77.3.3 GstTensorTrainerFrameworkInfo

```
typedef struct _GstTensorTrainerFrameworkInfo GstTensorTrainerFrameworkInfo
```

GstTensorTrainer's subplugin framework related information.

All the information is provided statically.

9.77.3.4 GstTensorTrainerProperties

```
typedef struct _GstTensorTrainerProperties GstTensorTrainerProperties
```

GstTensorTrainer's properties for neural network framework (internal data structure)

Internal data of GstTensorTrainer required by tensor_trainer's custom subplugin.

9.77.4 Enumeration Type Documentation

9.77.4.1 GstTensorTrainerEventType

```
enum GstTensorTrainerEventType
```

GstTensorTrainer's event type list.

Event types that subplugins must send to tensor_trainer

Enumerator

TRAINER_EVENT_EPOCH_COMPLETION	one epoch is complete in subplugin
TRAINER_EVENT_TRAINING_COMPLETION	training is complete in subplugin
TRAINER_EVENT_UNKNOWN	unknown event

Definition at line 67 of file nnstreamer_plugin_api_trainer.h.

9.77.5 Function Documentation

9.77.5.1 nnstreamer_trainer_exit()

```
int nnstreamer_trainer_exit (
            GstTensorTrainerFramework * ttsp )
```

Trainer's sub-plugin may call this to unregister itself.

Parameters

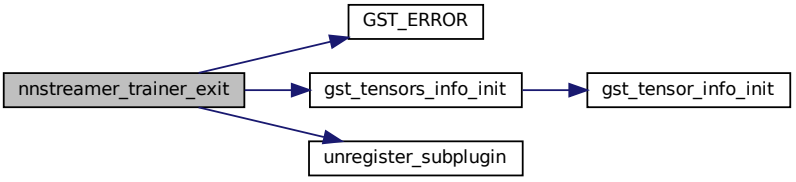
in	*ttsp*	tensor_trainer sub-plugin to be unregistered.

Returns

TRUE if unregistered. FALSE is failed.

Definition at line 1398 of file gsttensor_trainer.c.

Here is the call graph for this function:



9.77.5.2 nnstreamer_trainer_notify_event()

```
void nnstreamer_trainer_notify_event (
            GstTensorTrainerEventNotifier * notifier,
            GstTensorTrainerEventType type,
            void * data )
```

Trainer's sub-plugin call this to notify event.

Parameters

notifier	sub-plugin must send events to tensor_trainer with a notifier.
type	Event types that subplugins must send to tensor_trainer
data	Optional data for the event

Trainer's sub-plugin call this to notify event.

Parameters

in	*notifier*	event notifier, sub-plugin must send events with this.
in	*type*	event type

Definition at line 1425 of file gsttensor_trainer.c.

9.77.5.3 nnstreamer_trainer_probe()

```
int nnstreamer_trainer_probe (
            GstTensorTrainerFramework * ttsp )
```

Trainer's sub-plugin should call this function to register itself.

Parameters

in	*ttsp*	tensor_trainer sub-plugin to be registered.

Returns

TRUE if registered. FALSE is failed.

Note

Do not change the subplugins callbacks after probing the filter.

Parameters

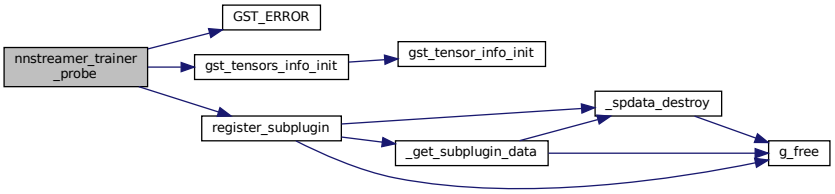
in	*ttsp*	tensor_trainer sub-plugin to be registered.

Returns

TRUE if registered. FALSE is failed or duplicated.

Definition at line 1371 of file gsttensor_trainer.c.

Here is the call graph for this function:

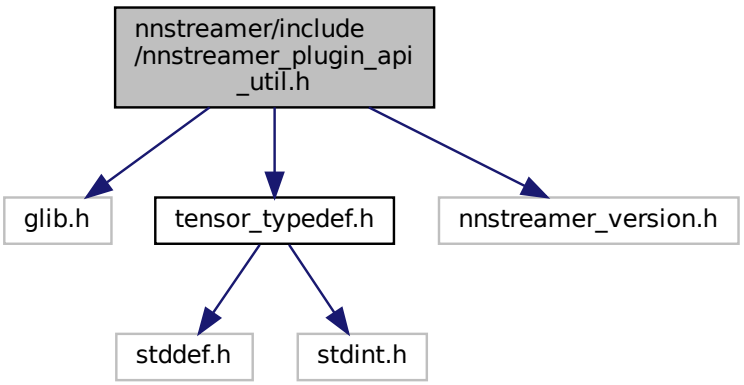


9.78 nnstreamer/include/nnstreamer_plugin_api_util.h File Reference

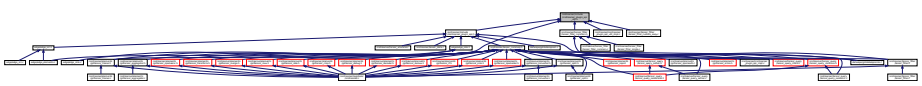
Optional/Additional NNStreamer APIs for sub-plugin writers. (No GStreamer dependency)

```
#include <glib.h>
#include <tensor_typedef.h>
#include <nnstreamer_version.h>
```

Include dependency graph for nnstreamer_plugin_api_util.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define _STR_NULL(str) ((str) ? (str) : "(NULL)")

  If the given string is NULL, print "(NULL)". Copied from `GST_STR_NULL`

- #define gst_tensors_config_is_static(c) ((c)->info.format == _NNS_TENSOR_FORMAT_STATIC)
  Macro to check stream format (static tensors for caps negotiation)
- #define gst_tensors_config_is_flexible(c) ((c)->info.format == _NNS_TENSOR_FORMAT_FLEXIBLE)
  Macro to check stream format (flexible tensors for caps negotiation)
- #define gst_tensors_config_is_sparse(c) ((c)->info.format == _NNS_TENSOR_FORMAT_SPARSE)
  Macro to check stream format (sparse tensors for caps negotiation)

Functions

- void gst_tensor_info_init (GstTensorInfo *info)
  Initialize the tensor info structure.
- void gst_tensor_info_free (GstTensorInfo *info)
  Free allocated data in tensor info structure.
- gsize gst_tensor_info_get_size (const GstTensorInfo *info)
  Get data size of single tensor.
- gboolean gst_tensor_info_validate (const GstTensorInfo *info)
  Check the tensor info is valid.
- gboolean gst_tensor_info_is_equal (const GstTensorInfo *i1, const GstTensorInfo *i2)
  Compare tensor info.
- void gst_tensor_info_copy_n (GstTensorInfo *dest, const GstTensorInfo *src, const guint n)
  Copy tensor info up to n elements.
- void gst_tensor_info_copy (GstTensorInfo *dest, const GstTensorInfo *src)
  Copy tensor info.
- gboolean gst_tensor_info_convert_to_meta (GstTensorInfo *info, GstTensorMetaInfo *meta)
  Convert GstTensorInfo structure to GstTensorMetaInfo.
- guint gst_tensor_info_get_rank (const GstTensorInfo *info)
  Get tensor rank.
- GstTensorInfo * gst_tensors_info_get_nth_info (GstTensorsInfo *info, guint index)
  Get the pointer of nth tensor information.
- void gst_tensors_info_init (GstTensorsInfo *info)
  Initialize the tensors info structure.
- void gst_tensors_info_free (GstTensorsInfo *info)
  Free allocated data in tensors info structure.
- gsize gst_tensors_info_get_size (const GstTensorsInfo *info, gint index)
  Get data size of single tensor.
- guint gst_tensors_info_parse_dimensions_string (GstTensorsInfo *info, const gchar *dim_string)
  Parse the string of dimensions.
- guint gst_tensors_info_parse_types_string (GstTensorsInfo *info, const gchar *type_string)
  Parse the string of types.
- guint gst_tensors_info_parse_names_string (GstTensorsInfo *info, const gchar *name_string)
  Parse the string of names.
- gchar * gst_tensors_info_get_dimensions_string (const GstTensorsInfo *info)
  Get the string of dimensions in tensors info.
- gchar * gst_tensors_info_get_rank_dimensions_string (const GstTensorsInfo *info, const unsigned int rank)
  Get the string of dimensions in tensors info and rank count.
- gchar * gst_tensors_info_get_types_string (const GstTensorsInfo *info)
  Get the string of types in tensors info.
- gchar * gst_tensors_info_get_names_string (const GstTensorsInfo *info)
  Get the string of tensor names in tensors info.
- gboolean gst_tensors_info_validate (const GstTensorsInfo *info)

Check the tensors info is valid.

- gboolean gst_tensors_info_is_equal (const GstTensorsInfo *i1, const GstTensorsInfo *i2)

  Compare tensors info.
- void gst_tensors_info_copy (GstTensorsInfo *dest, const GstTensorsInfo *src)

  Copy tensor info.
- gchar * gst_tensors_info_to_string (const GstTensorsInfo *info)

  GstTensorsInfo represented as a string. Caller should free it.
- void gst_tensors_config_init (GstTensorsConfig *config)

  Initialize the tensors config info structure (for other/tensors)
- void gst_tensors_config_free (GstTensorsConfig *config)

  Free allocated data in tensors config structure.
- gboolean gst_tensors_config_validate (const GstTensorsConfig *config)

  Check the tensors are all configured (for other/tensors)
- gboolean gst_tensors_config_is_equal (const GstTensorsConfig *c1, const GstTensorsConfig *c2)

  Compare tensor config info (for other/tensors)
- void gst_tensors_config_copy (GstTensorsConfig *dest, const GstTensorsConfig *src)

  Copy tensors config.
- gchar * gst_tensors_config_to_string (const GstTensorsConfig *config)

  Tensor config represented as a string. Caller should free it.
- gboolean gst_tensor_dimension_is_valid (const tensor_dim dim)

  Check the tensor dimension is valid.
- gboolean gst_tensor_dimension_is_equal (const tensor_dim dim1, const tensor_dim dim2)

  Compare the tensor dimension.
- guint gst_tensor_dimension_get_rank (const tensor_dim dim)

  Get the rank of tensor dimension.
- guint gst_tensor_dimension_get_min_rank (const tensor_dim dim)

  Get the minimum rank of tensor dimension.
- guint gst_tensor_parse_dimension (const gchar *dimstr, tensor_dim dim)

  Parse tensor dimension parameter string.
- gchar * gst_tensor_get_dimension_string (const tensor_dim dim)

  Get dimension string from given tensor dimension.
- gchar * gst_tensor_get_rank_dimension_string (const tensor_dim dim, const unsigned int rank)

  Get dimension string from given tensor dimension and rank count.
- gboolean gst_tensor_dimension_string_is_equal (const gchar *dimstr1, const gchar *dimstr2)

  Compare dimension strings.
- gulong gst_tensor_get_element_count (const tensor_dim dim)

  Count the number of elements of a tensor.
- gsize gst_tensor_get_element_size (tensor_type type)

  Get element size of tensor type (byte per element)
- tensor_type gst_tensor_get_type (const gchar *typestr)

  Get tensor type from string input.
- const gchar * gst_tensor_get_type_string (tensor_type type)

  Get type string of tensor type.
- tensor_format gst_tensor_get_format (const gchar *format_str)

  Get tensor format from string input.
- const gchar * gst_tensor_get_format_string (tensor_format format)

  Get tensor format string.
- gint find_key_strv (const gchar **strv, const gchar *key)

  Find the index value of the given key string array.
- void gst_tensor_meta_info_init (GstTensorMetaInfo *meta)

  Initialize the tensor meta info structure.

- void gst_tensor_meta_info_get_version (GstTensorMetaInfo *meta, guint *major, guint *minor)

  Get the version of tensor meta.
- gboolean gst_tensor_meta_info_validate (GstTensorMetaInfo *meta)

  Check the meta info is valid.
- gsize gst_tensor_meta_info_get_header_size (GstTensorMetaInfo *meta)

  Get the header size to handle a tensor meta.
- gsize gst_tensor_meta_info_get_data_size (GstTensorMetaInfo *meta)

  Get the data size calculated from tensor meta.
- gboolean gst_tensor_meta_info_update_header (GstTensorMetaInfo *meta, gpointer header)

  Update header from tensor meta.
- gboolean gst_tensor_meta_info_parse_header (GstTensorMetaInfo *meta, gpointer header)

  Parse header and fill the tensor meta.
- gboolean gst_tensor_meta_info_convert (GstTensorMetaInfo *meta, GstTensorInfo *info)

  Convert GstTensorMetaInfo structure to GstTensorInfo.
- gchar * nnstreamer_version_string (void)

  Get the version of NNStreamer.
- void nnstreamer_version_fetch (guint *major, guint *minor, guint *micro)

  Get the version of NNStreamer (int, divided).

9.78.1 Detailed Description

Optional/Additional NNStreamer APIs for sub-plugin writers. (No GStreamer dependency)

NNStreamer Common API Header for sub-plugin writers Copyright (C) 2022 Gichan Jang gichan2.↩ jang@samsung.com

Date

28 Jan 2022

See also

https://github.com/nnstreamer/nnstreamer

Author

Gichan Jang gichan2.jang@samsung.com

Bug No known bugs except for NYI items

9.78.2 Macro Definition Documentation

9.78.2.1 _STR_NULL

```
#define _STR_NULL(
              str ) ((str) ?  (str) :  "(NULL)")
```

If the given string is NULL, print "(NULL)". Copied from GST_STR_NULL

Definition at line 26 of file nnstreamer_plugin_api_util.h.

9.78.2.2 gst_tensors_config_is_flexible

```
#define gst_tensors_config_is_flexible(
              c ) ((c)->info.format == _NNS_TENSOR_FORMAT_FLEXIBLE)
```

Macro to check stream format (flexible tensors for caps negotiation)

Definition at line 279 of file nnstreamer_plugin_api_util.h.

9.78.2.3 gst_tensors_config_is_sparse

```
#define gst_tensors_config_is_sparse(
              c ) ((c)->info.format == _NNS_TENSOR_FORMAT_SPARSE)
```

Macro to check stream format (sparse tensors for caps negotiation)

Definition at line 284 of file nnstreamer_plugin_api_util.h.

9.78.2.4 gst_tensors_config_is_static

```
#define gst_tensors_config_is_static(
              c ) ((c)->info.format == _NNS_TENSOR_FORMAT_STATIC)
```

Macro to check stream format (static tensors for caps negotiation)

Definition at line 274 of file nnstreamer_plugin_api_util.h.

9.78.3 Function Documentation

9.78.3.1 find_key_strv()

```
gint find_key_strv (
             const gchar ** strv,
             const gchar * key )
```

Find the index value of the given key string array.

Returns

Corresponding index

Parameters

strv	Null terminated array of gchar *
key	The key string value

Returns

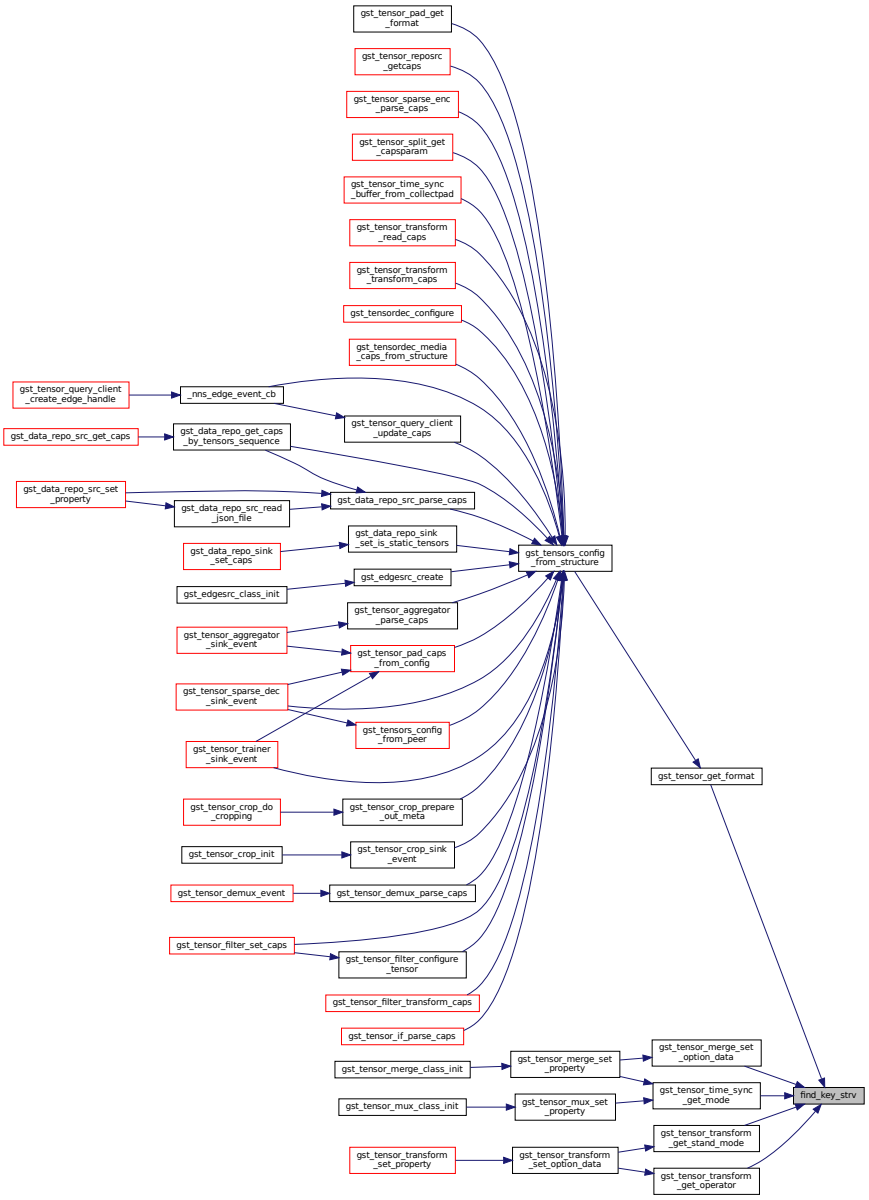
Corresponding index. Returns -1 if not found.

Parameters

strv	Null terminated array of gchar *
key	The key string value

Definition at line 1586 of file nnstreamer_plugin_api_util_impl.c.

Here is the caller graph for this function:



9.78.3.2 gst_tensor_dimension_get_min_rank()

```
guint gst_tensor_dimension_get_min_rank (
            const tensor_dim dim )
```

Get the minimum rank of tensor dimension.

The C-arrays with dim 4:4:4 and 4:4:4:1 have same data. In this case, this function returns min rank 3.

Parameters

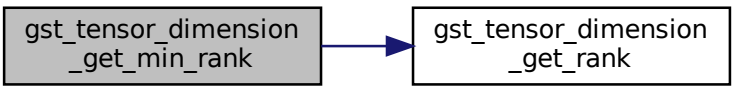
dim	tensor dimension.

Returns

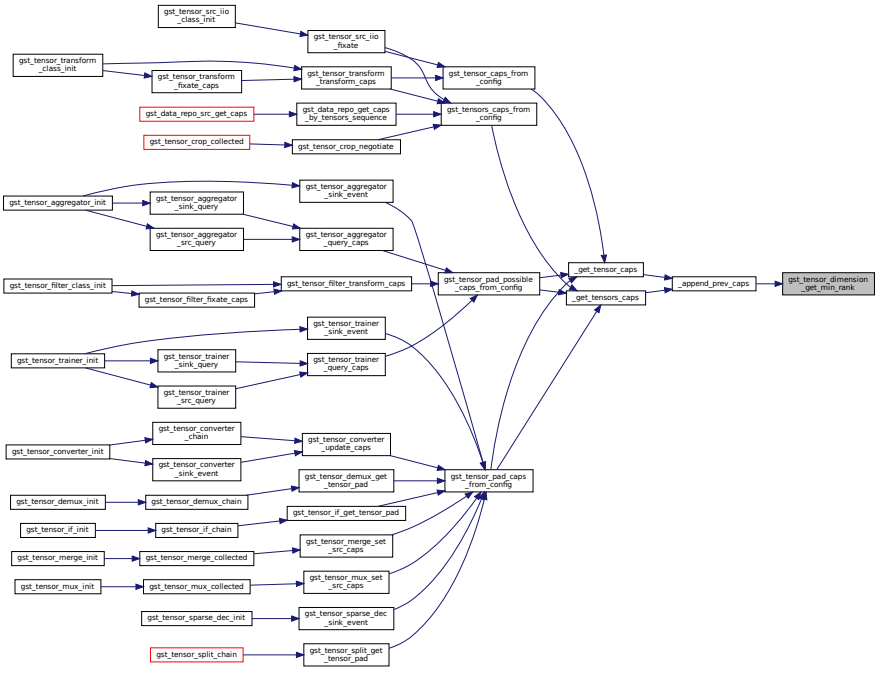
tensor rank (Minimum rank is 1 if given dimension is valid)

Definition at line 1017 of file nnstreamer_plugin_api_util_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.78.3.3 gst_tensor_dimension_get_rank()

```
guint gst_tensor_dimension_get_rank (
            const tensor_dim dim )
```

Get the rank of tensor dimension.

Parameters

dim	tensor dimension.

Returns

tensor rank (Minimum rank is 1 if given info is valid)

Parameters

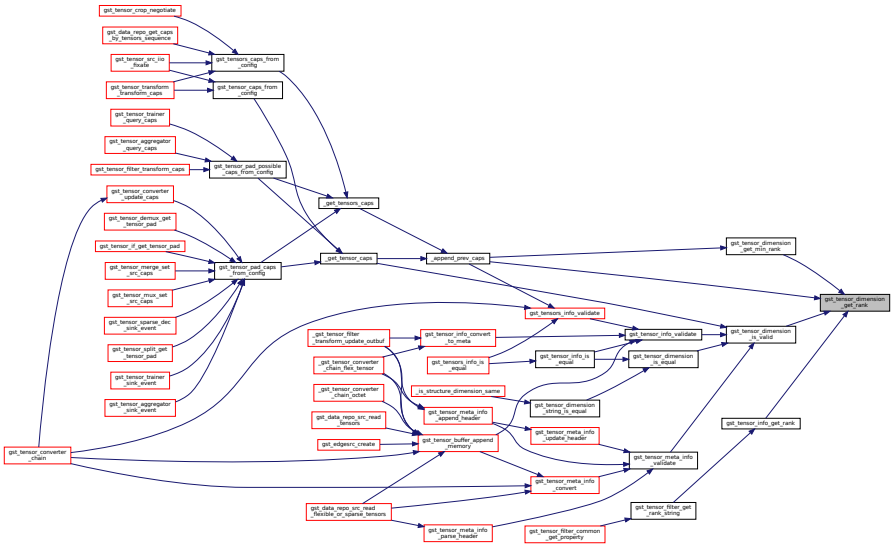
dim	tensor dimension.

Returns

tensor rank (Minimum rank is 1 if given dimension is valid)

Definition at line 998 of file nnstreamer_plugin_api_util_impl.c.

Here is the caller graph for this function:



9.78.3.4 gst_tensor_dimension_is_equal()

```
gboolean gst_tensor_dimension_is_equal (
            const tensor_dim dim1,
            const tensor_dim dim2 )
```

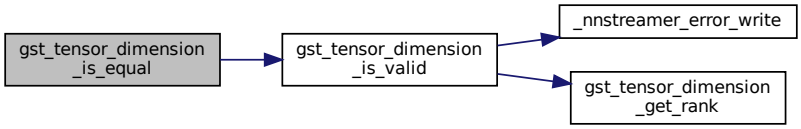
Compare the tensor dimension.

Returns

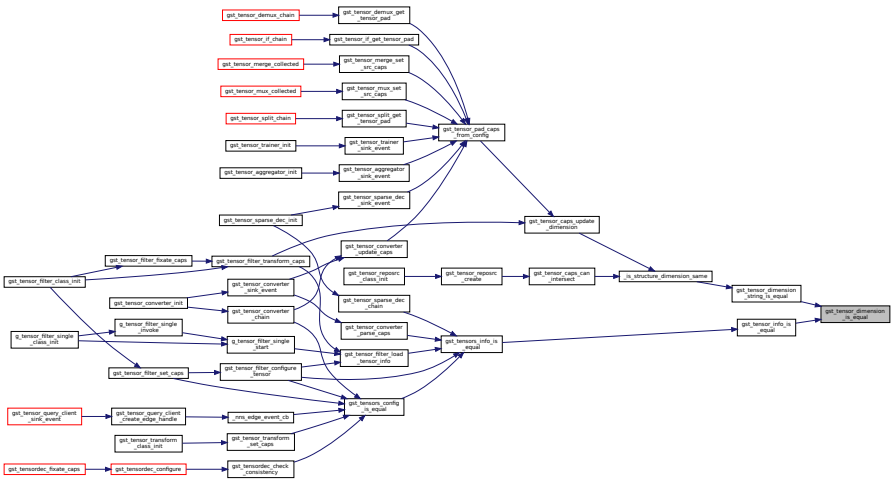
TRUE if given tensors have same dimension.

Definition at line 972 of file nnstreamer_plugin_api_util_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.78.3.5 gst_tensor_dimension_is_valid()

```
gboolean gst_tensor_dimension_is_valid (
            const tensor_dim dim )
```

Check the tensor dimension is valid.

Parameters

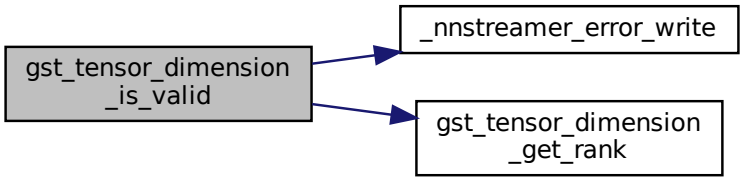
dim	tensor dimension

Returns

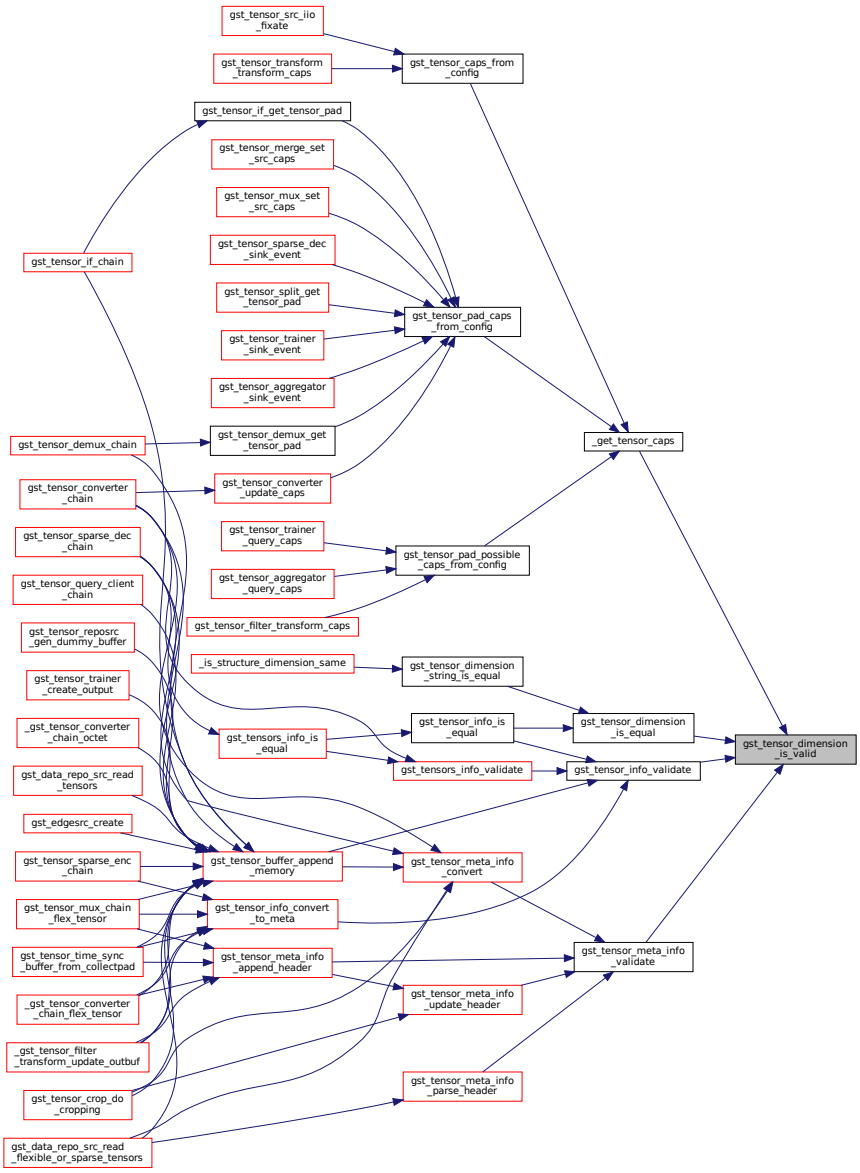
TRUE if dimension is valid

Definition at line 940 of file nnstreamer_plugin_api_util_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.78.3.6 gst_tensor_dimension_string_is_equal()

```
gboolean gst_tensor_dimension_string_is_equal (
            const gchar * dimstr1,
            const gchar * dimstr2 )
```

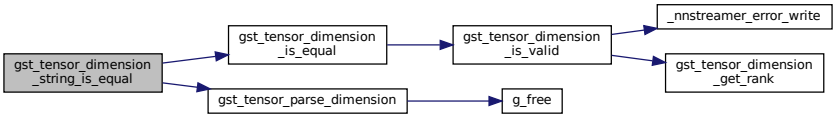
Compare dimension strings.

Returns

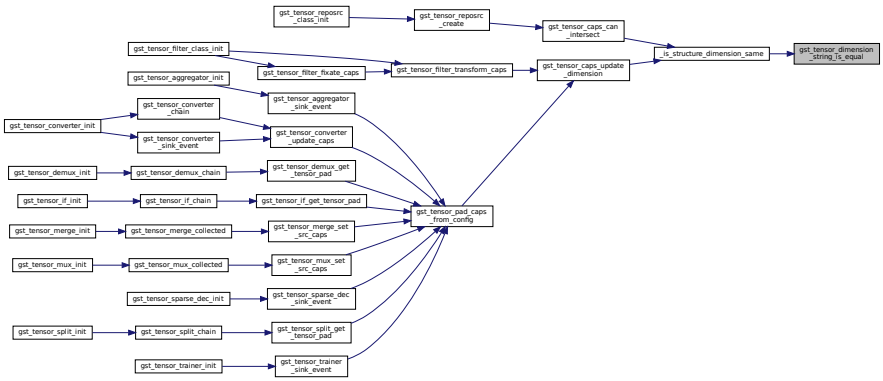
TRUE if equal, FALSE if given dimension strings are invalid or not equal.

Definition at line 1140 of file nnstreamer_plugin_api_util_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.78.3.7 gst_tensor_get_dimension_string()

```
gchar* gst_tensor_get_dimension_string (
            const tensor_dim dim )
```

Get dimension string from given tensor dimension.

Parameters

dim	tensor dimension

Returns

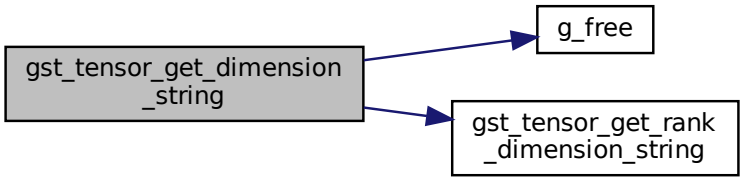
Formatted string of given dimension (d1:d2:d3:...:d15:d16).

Note

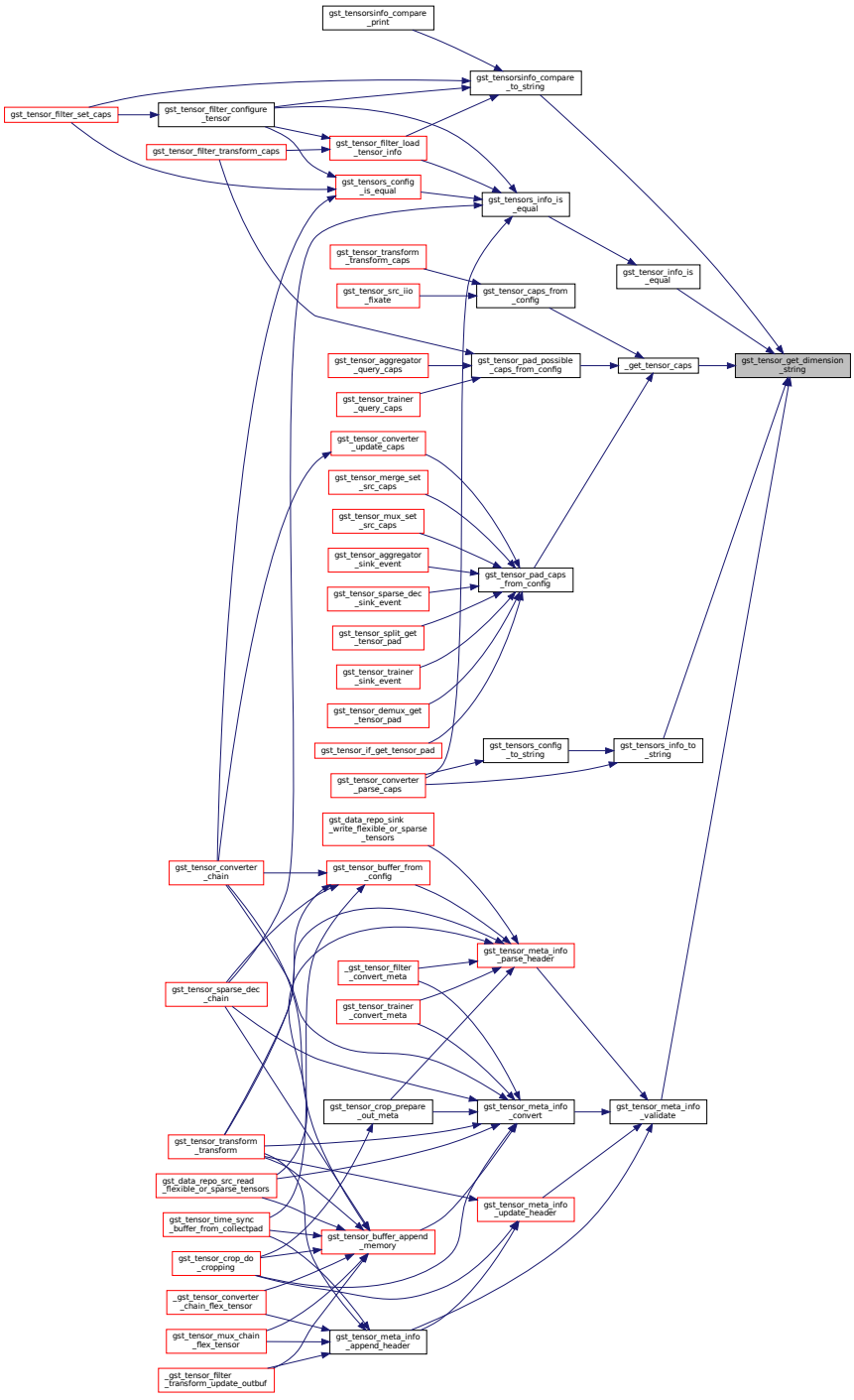
The returned value should be freed with g_free()

Definition at line 1083 of file nnstreamer_plugin_api_util_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.78.3.8 gst_tensor_get_element_count()

```
gulong gst_tensor_get_element_count (
            const tensor_dim dim )
```

Count the number of elements of a tensor.

Returns

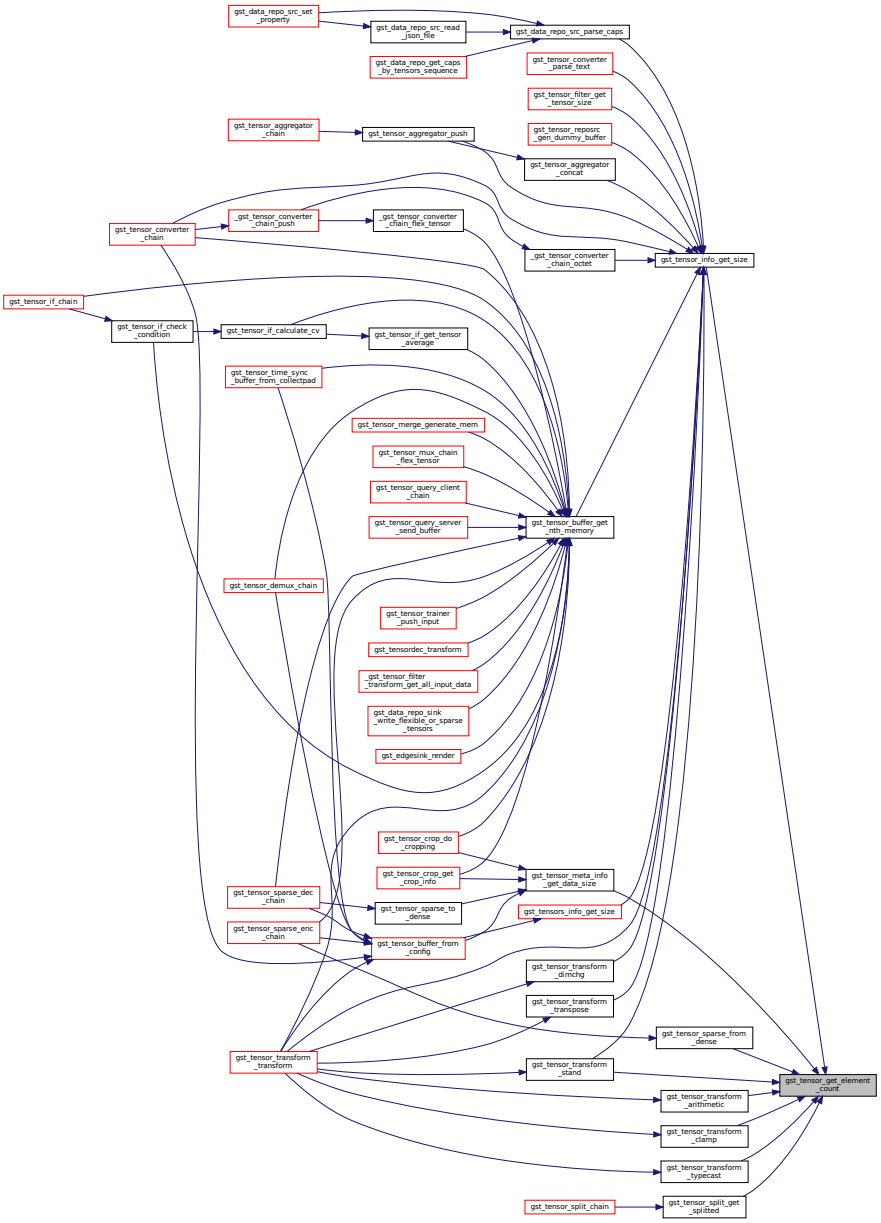
The number of elements. 0 if error.

Parameters

dim	The tensor dimension

Definition at line 1186 of file nnstreamer_plugin_api_util_impl.c.

Here is the caller graph for this function:



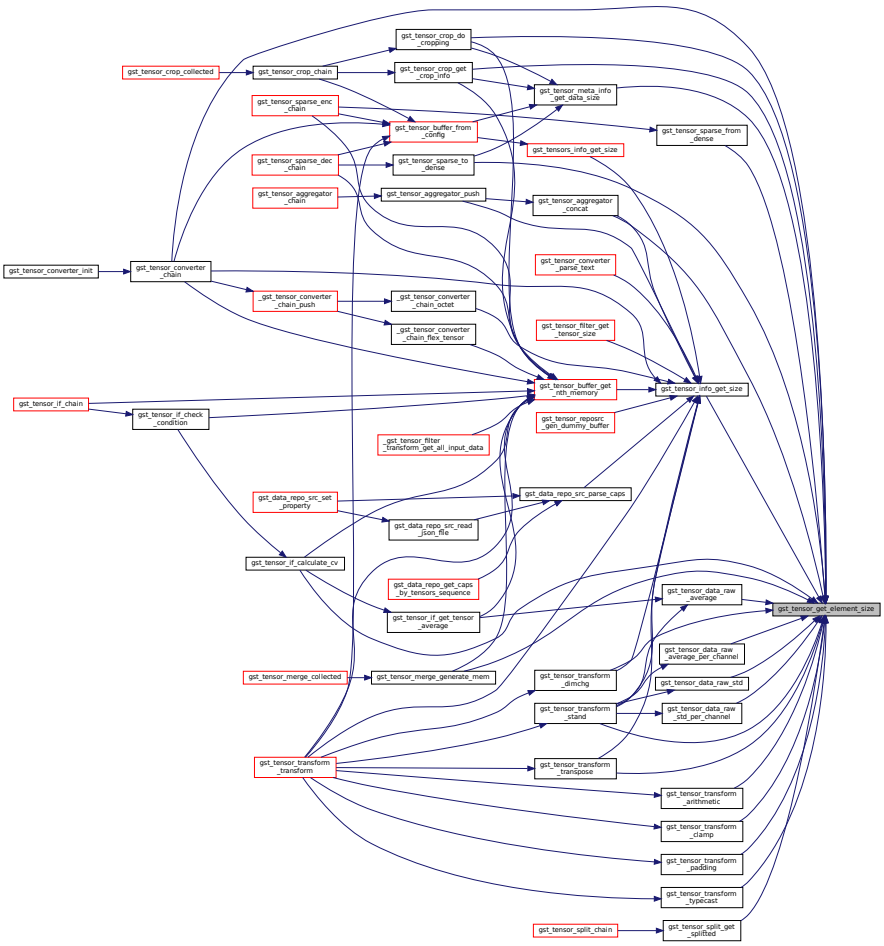
9.78.3.9 gst_tensor_get_element_size()

```
gsize gst_tensor_get_element_size (
            tensor_type type )
```

Get element size of tensor type (byte per element)

Definition at line 1205 of file nnstreamer_plugin_api_util_impl.c.

Here is the caller graph for this function:



9.78.3.10 gst_tensor_get_format()

```
tensor_format gst_tensor_get_format (
            const gchar * format_str )
```

Get tensor format from string input.

Parameters

format_str	The string format name, supposed to be one of tensor_format_name[].

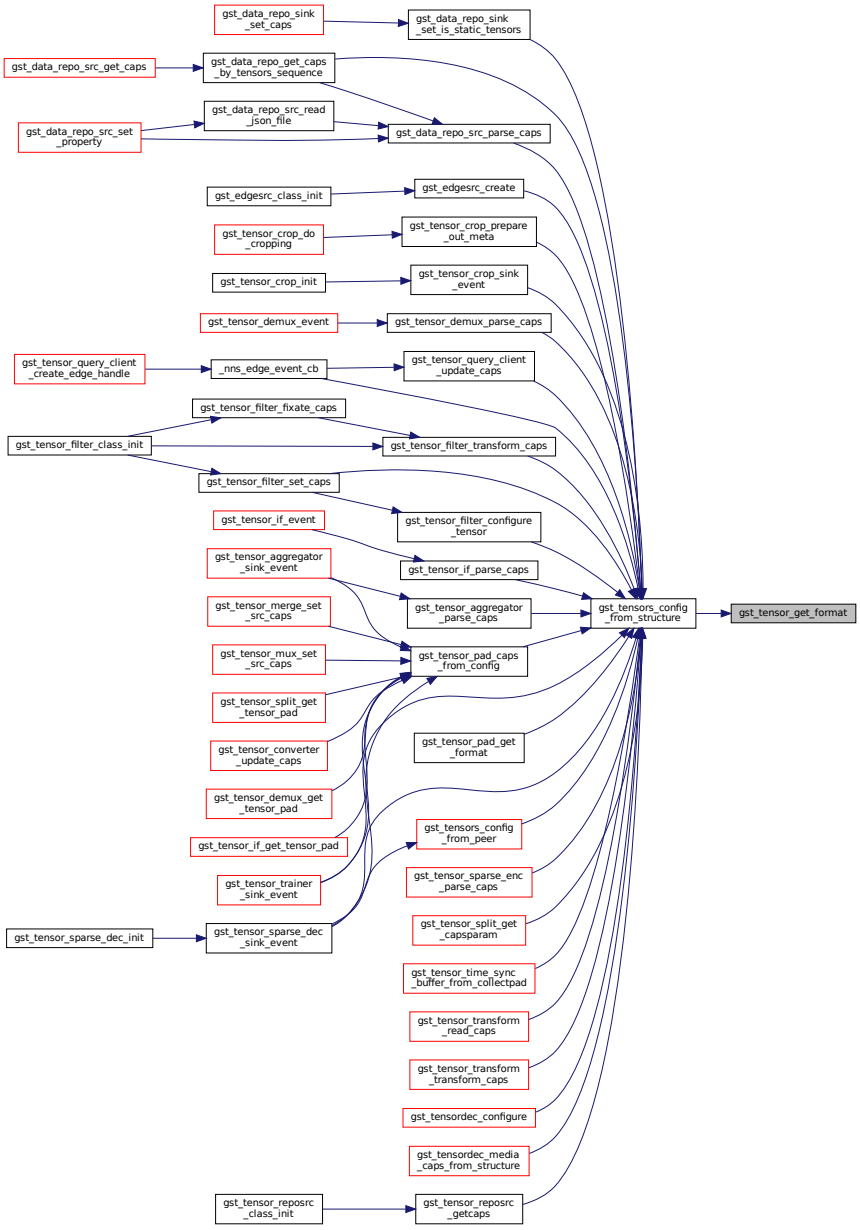
Returns

Corresponding tensor_format. _NNS_TENSOR_FORMAT_END if unrecognized value is there.

Definition at line 1309 of file nnstreamer_plugin_api_util_impl.c.

Here is the call graph for this function:

Here is the caller graph for this function:



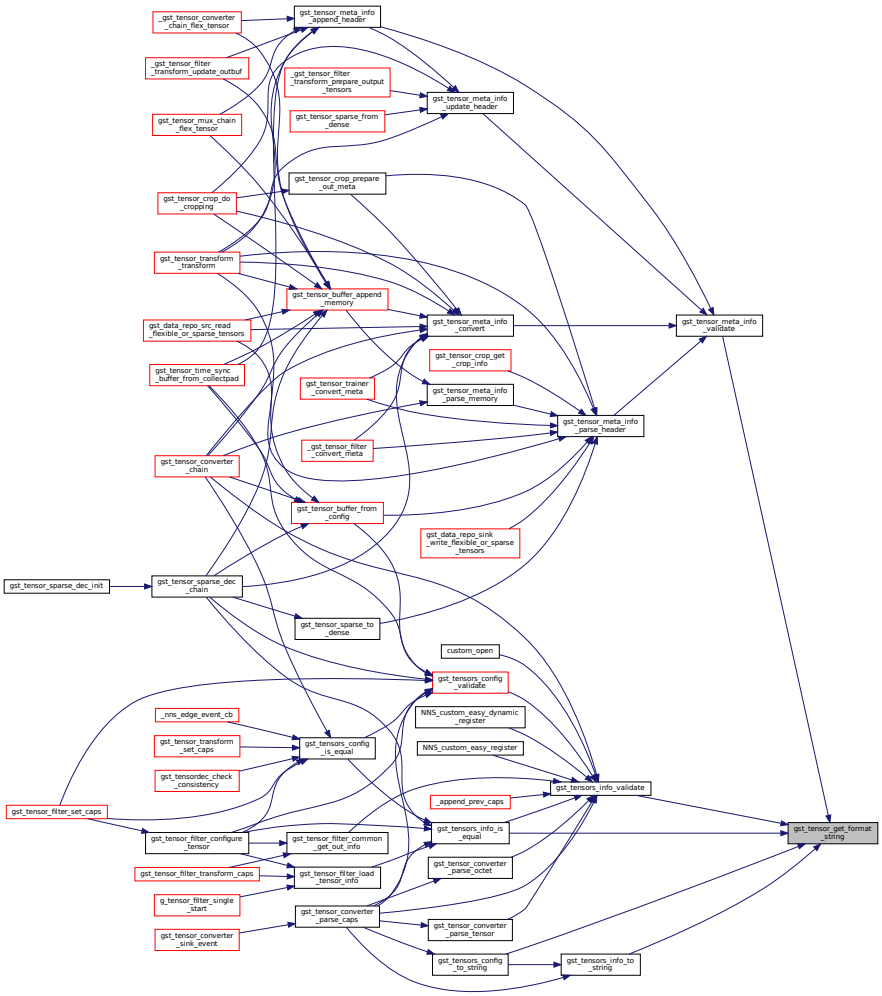
9.78.3.11 gst_tensor_get_format_string()

```
const gchar* gst_tensor_get_format_string (
            tensor_format format )
```

Get tensor format string.

Definition at line 1325 of file nnstreamer_plugin_api_util_impl.c.

Here is the caller graph for this function:



9.78.3.12 gst_tensor_get_rank_dimension_string()

```
gchar* gst_tensor_get_rank_dimension_string (
            const tensor_dim dim,
            const unsigned int rank )
```

Get dimension string from given tensor dimension and rank count.

Parameters

dim	tensor dimension
rank	rank count of given tensor dimension

Returns

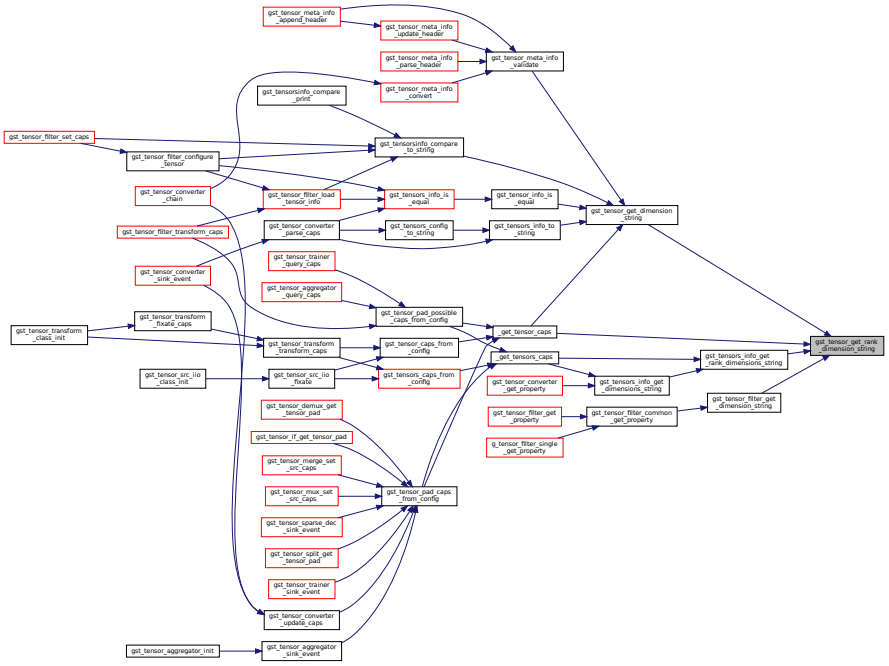
Formatted string of given dimension

Note

If rank count is 3, then returned string is 'd1:d2:d3`. The returned value should be freed with g_free().

Definition at line 1107 of file nnstreamer_plugin_api_util_impl.c.

Here is the caller graph for this function:



9.78.3.13 gst_tensor_get_type()

```
tensor_type gst_tensor_get_type (
            const gchar * typestr )
```

Get tensor type from string input.

Returns

Corresponding tensor_type. _NNS_END if unrecognized value is there.

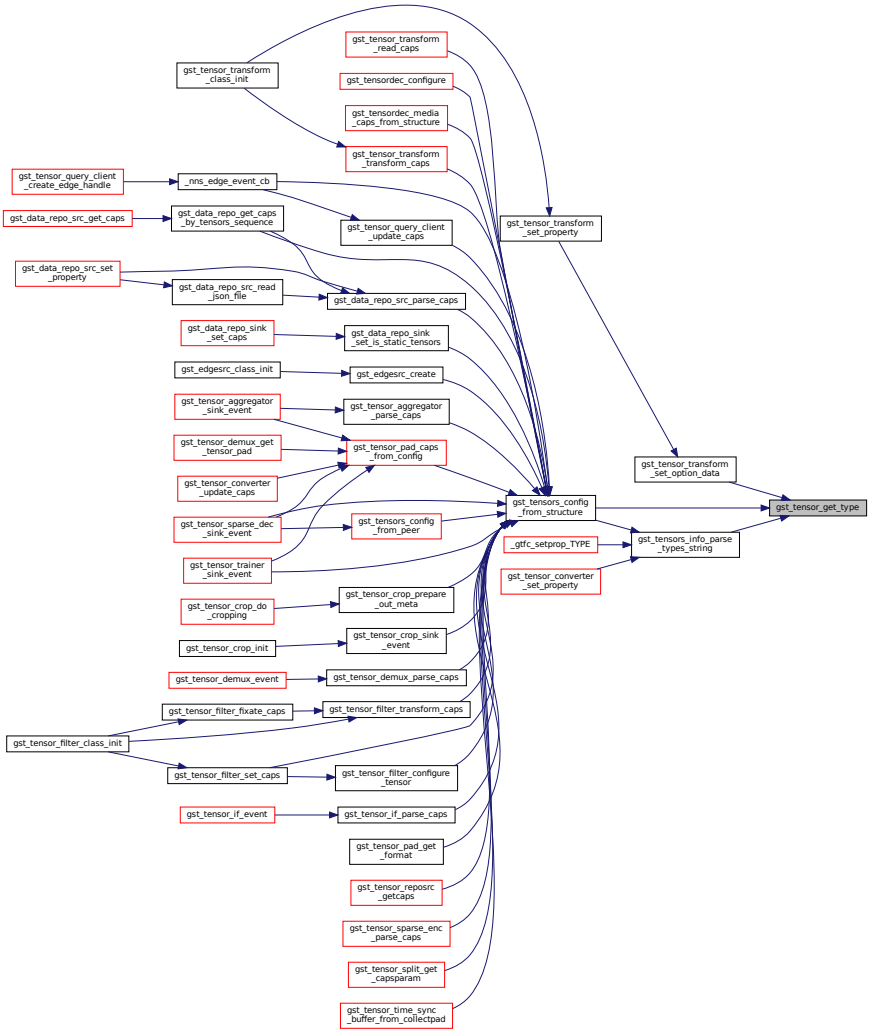
Parameters

typestr	The string type name, supposed to be one of tensor_element_typename[]

Definition at line 1218 of file nnstreamer_plugin_api_util_impl.c.

Here is the call graph for this function:

Here is the caller graph for this function:



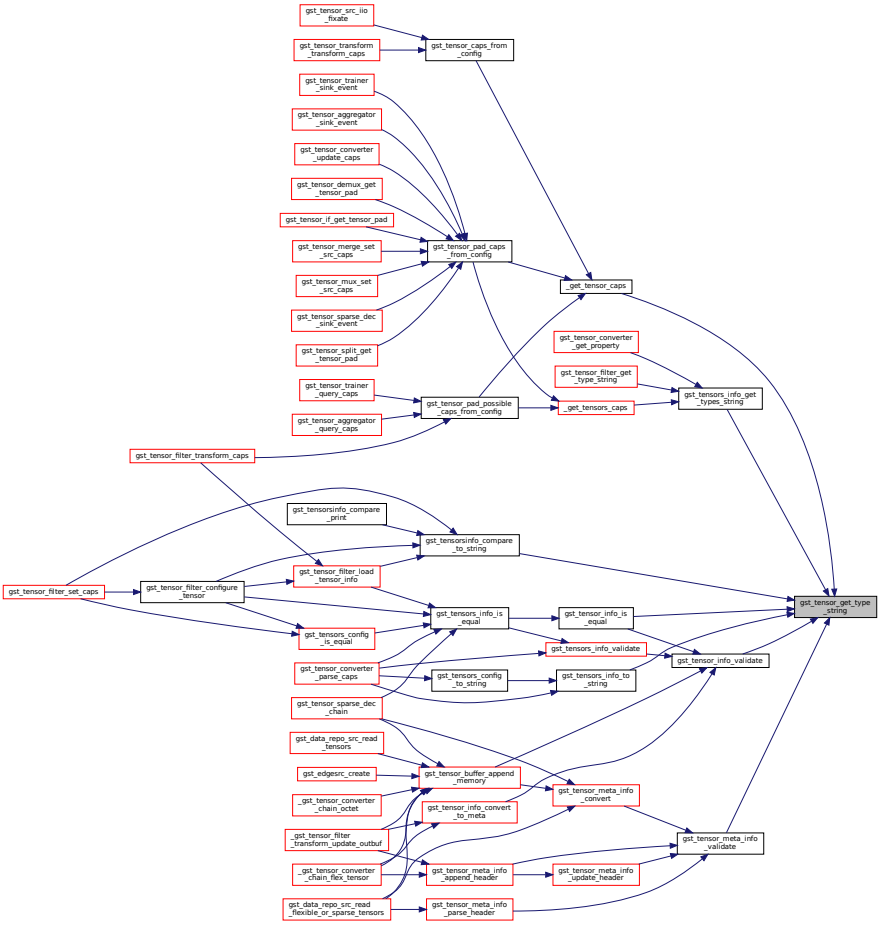
9.78.3.14 gst_tensor_get_type_string()

```
const gchar* gst_tensor_get_type_string (
            tensor_type type )
```

Get type string of tensor type.

Definition at line 1296 of file nnstreamer_plugin_api_util_impl.c.

Here is the caller graph for this function:



9.78.3.15 gst_tensor_info_convert_to_meta()

```
gboolean gst_tensor_info_convert_to_meta (
            GstTensorInfo * info,
            GstTensorMetaInfo * meta )
```

Convert GstTensorInfo structure to GstTensorMetaInfo.

Parameters

in	*info*	GstTensorInfo to be converted
out	*meta*	tensor meta structure to be filled

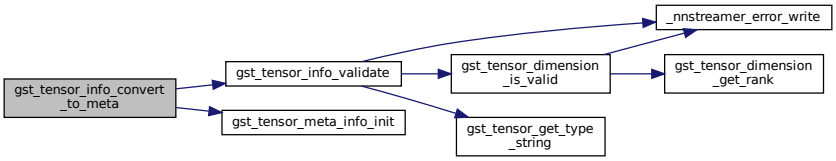
Returns

TRUE if successfully set the meta

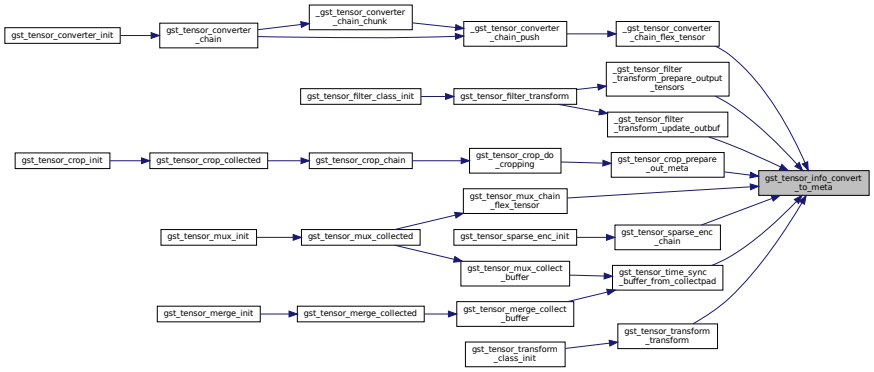
Todo handle rank from info.dimension

Definition at line 260 of file nnstreamer_plugin_api_util_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.78.3.16 gst_tensor_info_copy()

```
void gst_tensor_info_copy (
            GstTensorInfo * dest,
            const GstTensorInfo * src )
```

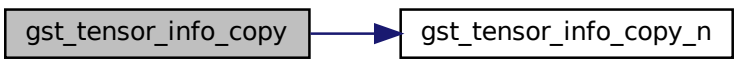
Copy tensor info.

Note

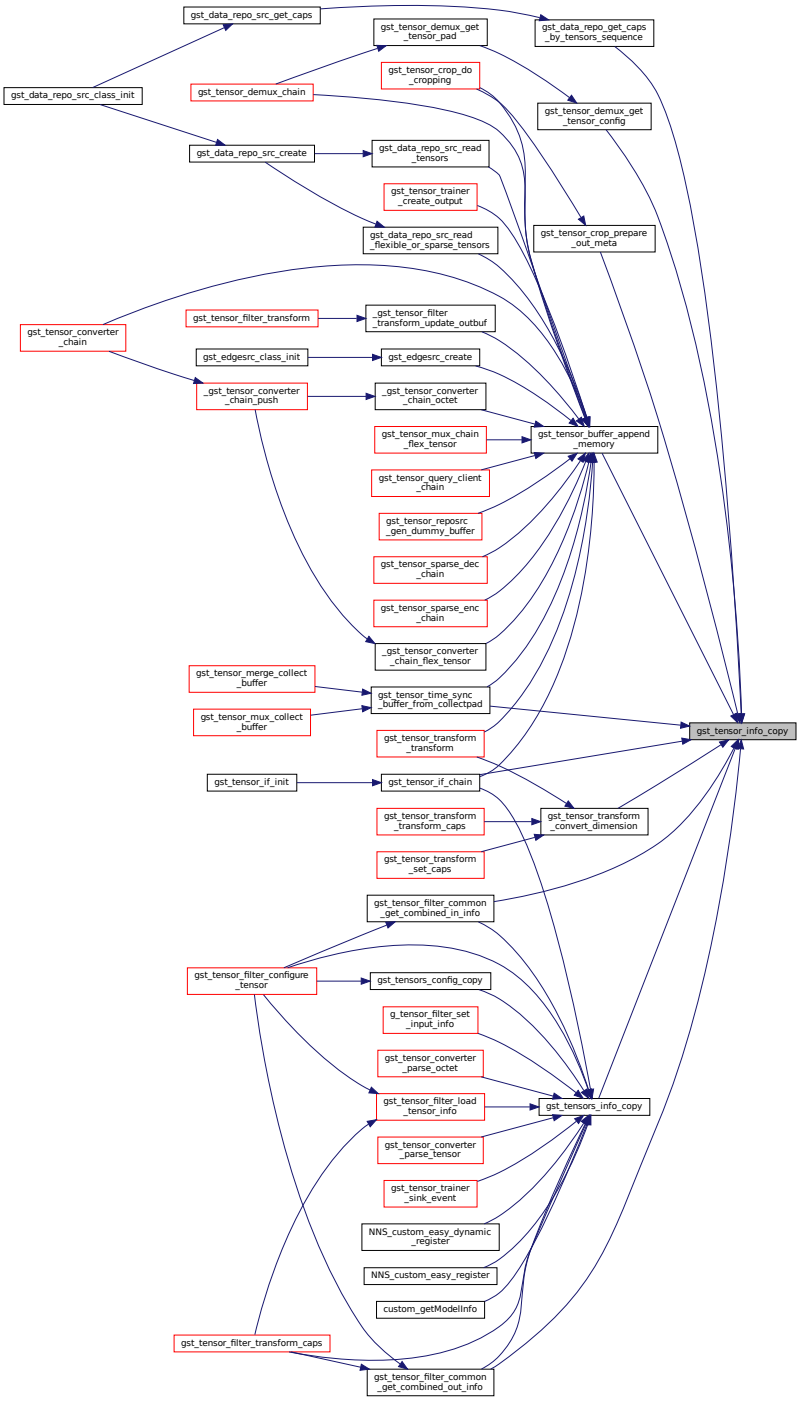
Copied info should be freed with gst_tensor_info_free()

Definition at line 248 of file nnstreamer_plugin_api_util_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.78.3.17 gst_tensor_info_copy_n()

```
void gst_tensor_info_copy_n (
            GstTensorInfo * dest,
```

```
const GstTensorInfo * src,
const guint n )
```

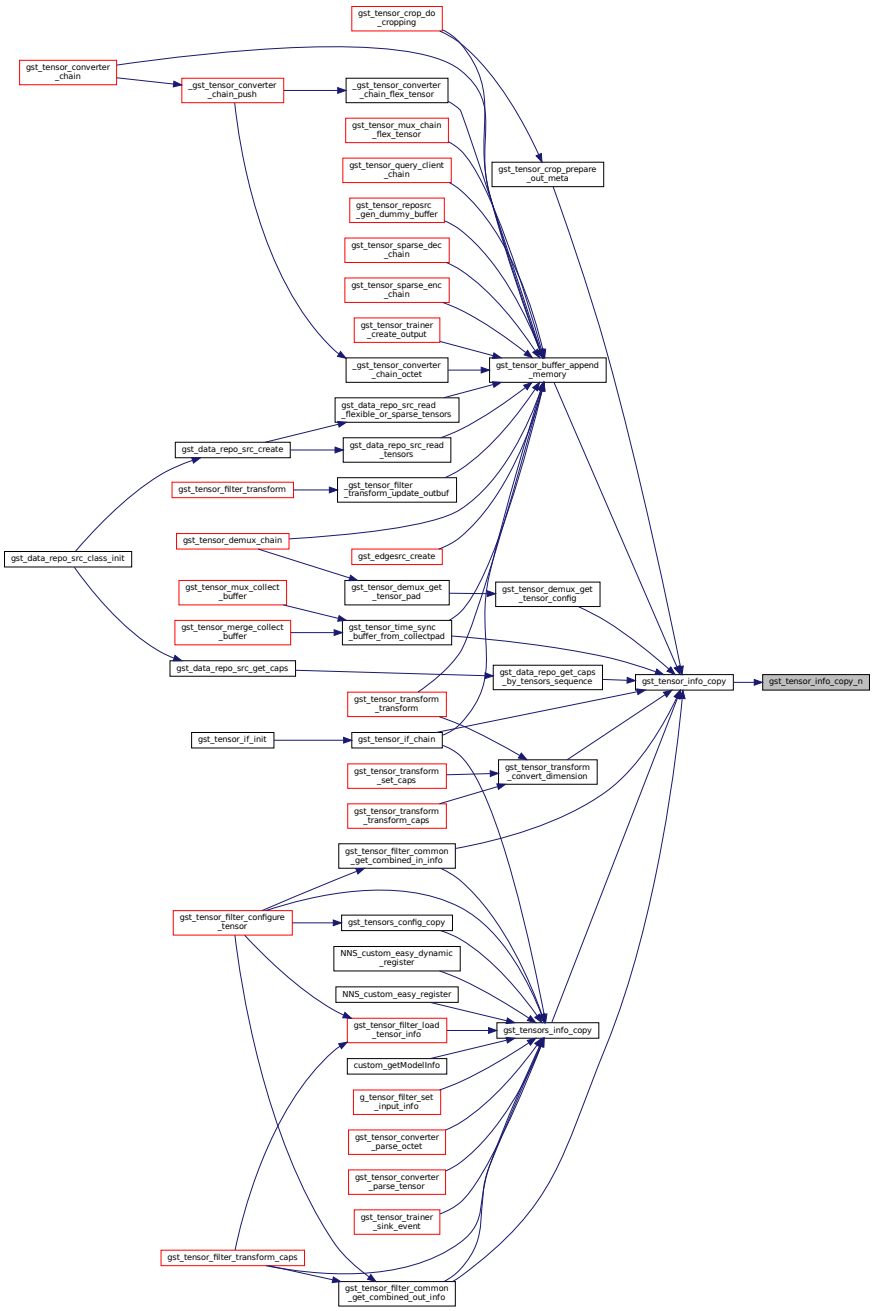
Copy tensor info up to n elements.

Note

Copied info should be freed with gst_tensor_info_free()

Definition at line 227 of file nnstreamer_plugin_api_util_impl.c.

Here is the caller graph for this function:



9.78.3.18 gst_tensor_info_free()

```
void gst_tensor_info_free (
            GstTensorInfo * info )
```

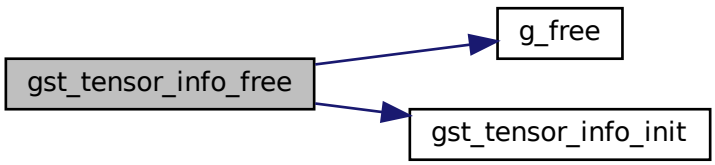
Free allocated data in tensor info structure.

Parameters

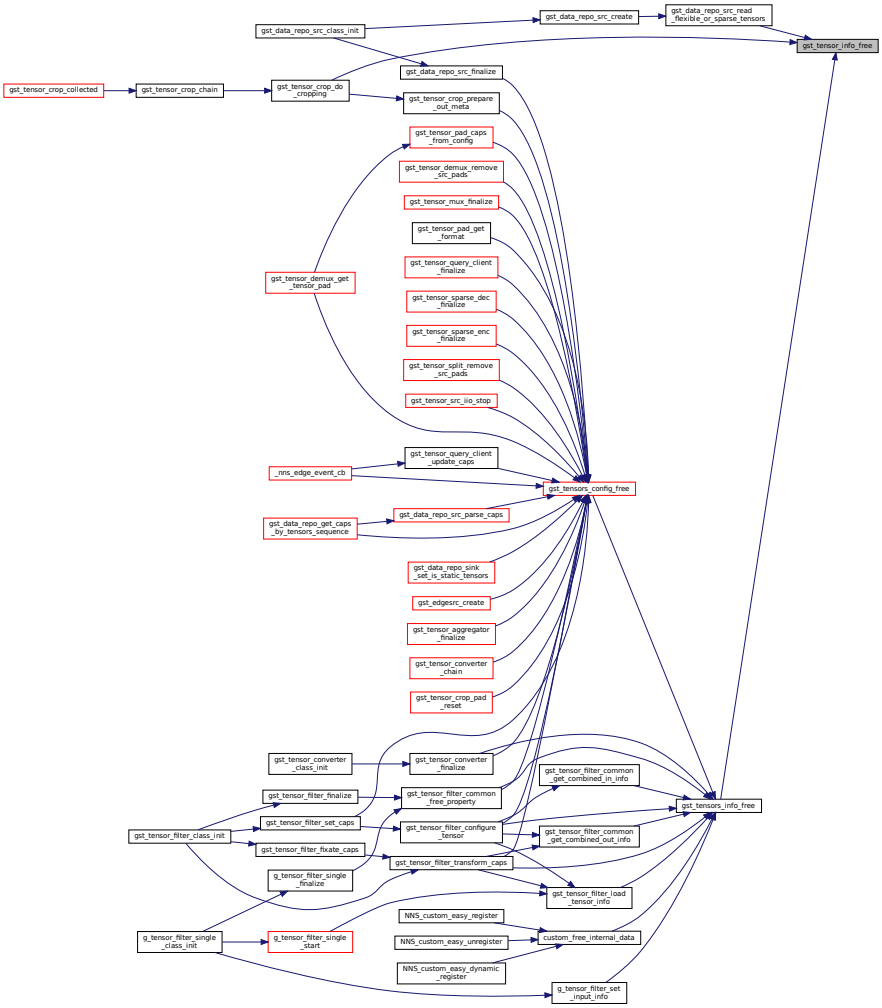
info	tensor info structure

Definition at line 140 of file nnstreamer_plugin_api_util_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.78.3.19 gst_tensor_info_get_rank()

```
guint gst_tensor_info_get_rank (
            const GstTensorInfo * info )
```

Get tensor rank.

Parameters

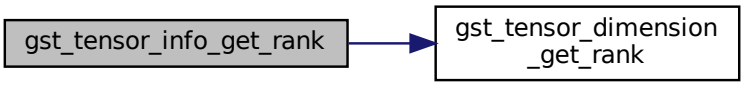
info	tensor info structure

Returns

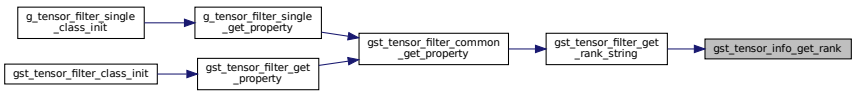
tensor rank (Minimum rank is 1 if given info is valid)

Definition at line 285 of file nnstreamer_plugin_api_util_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.78.3.20 gst_tensor_info_get_size()

```
gsize gst_tensor_info_get_size (
            const GstTensorInfo * info )
```

Get data size of single tensor.

Parameters

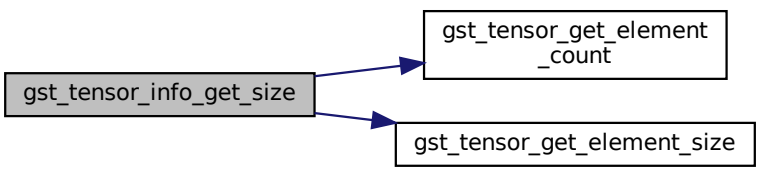
info	tensor info structure

Returns

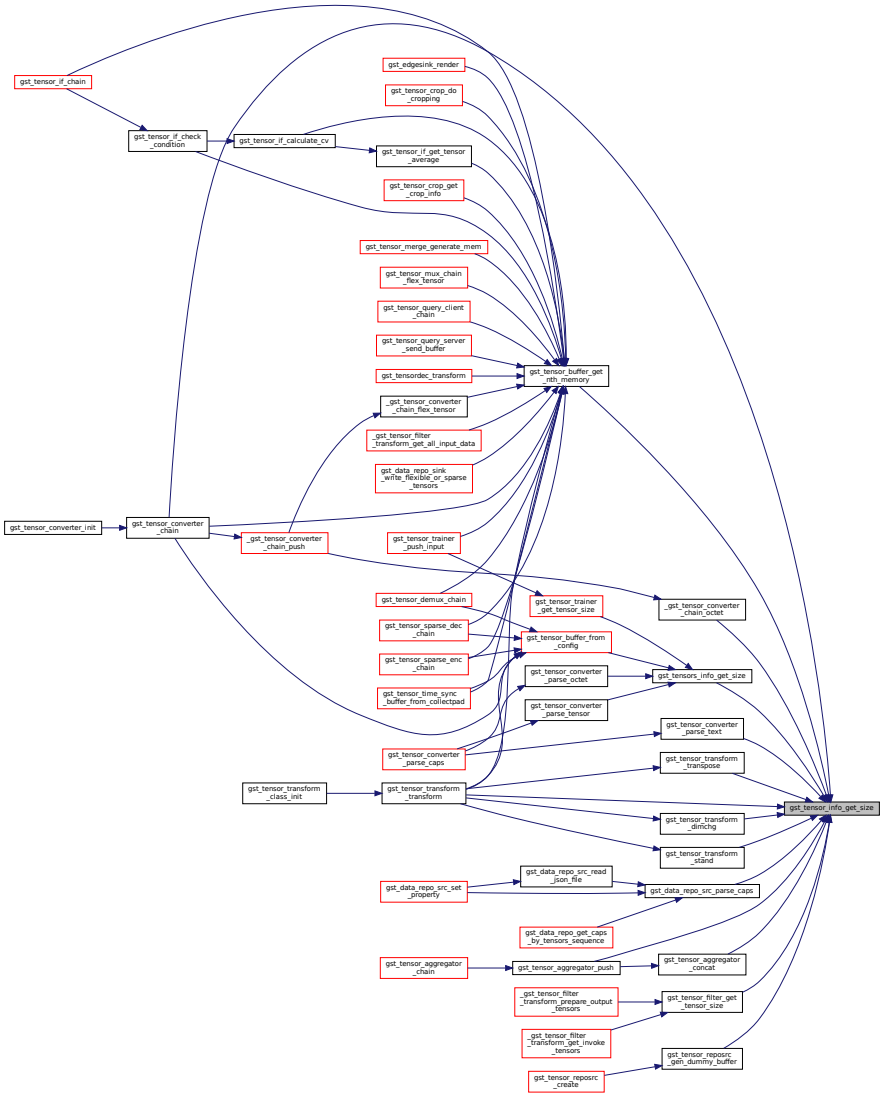
data size

Definition at line 156 of file nnstreamer_plugin_api_util_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.78.3.21 gst_tensor_info_init()

```
void gst_tensor_info_init (
            GstTensorInfo * info )
```

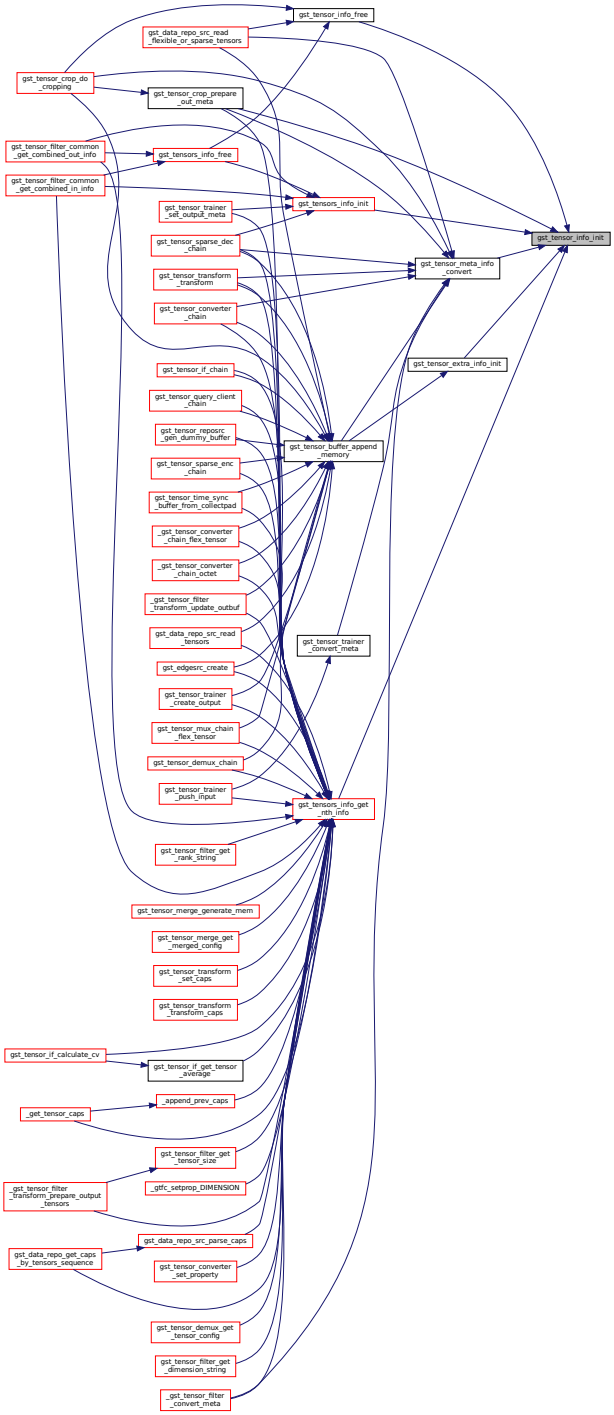
Initialize the tensor info structure.

Parameters

info	tensor info structure to be initialized

Definition at line 121 of file nnstreamer_plugin_api_util_impl.c.

Here is the caller graph for this function:



9.78.3.22 gst_tensor_info_is_equal()

```
gboolean gst_tensor_info_is_equal (
            const GstTensorInfo * i1,
            const GstTensorInfo * i2 )
```

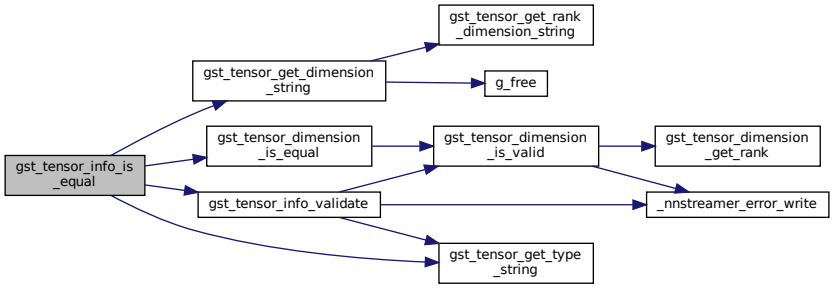
Compare tensor info.

Returns

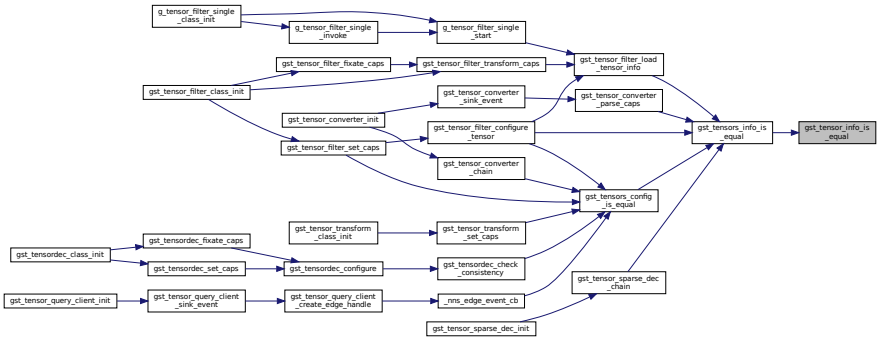
TRUE if equal, FALSE if given tensor infos are invalid or not equal.

Definition at line 197 of file nnstreamer_plugin_api_util_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.78.3.23 gst_tensor_info_validate()

```
gboolean gst_tensor_info_validate (
            const GstTensorInfo * info )
```

Check the tensor info is valid.

Parameters

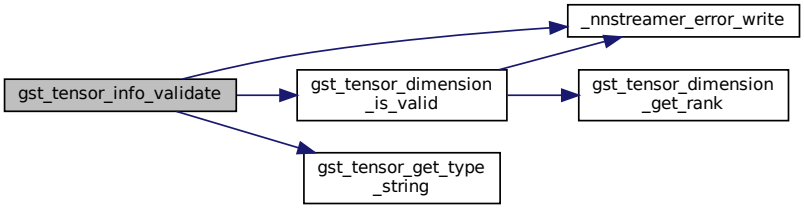
info	tensor info structure

Returns

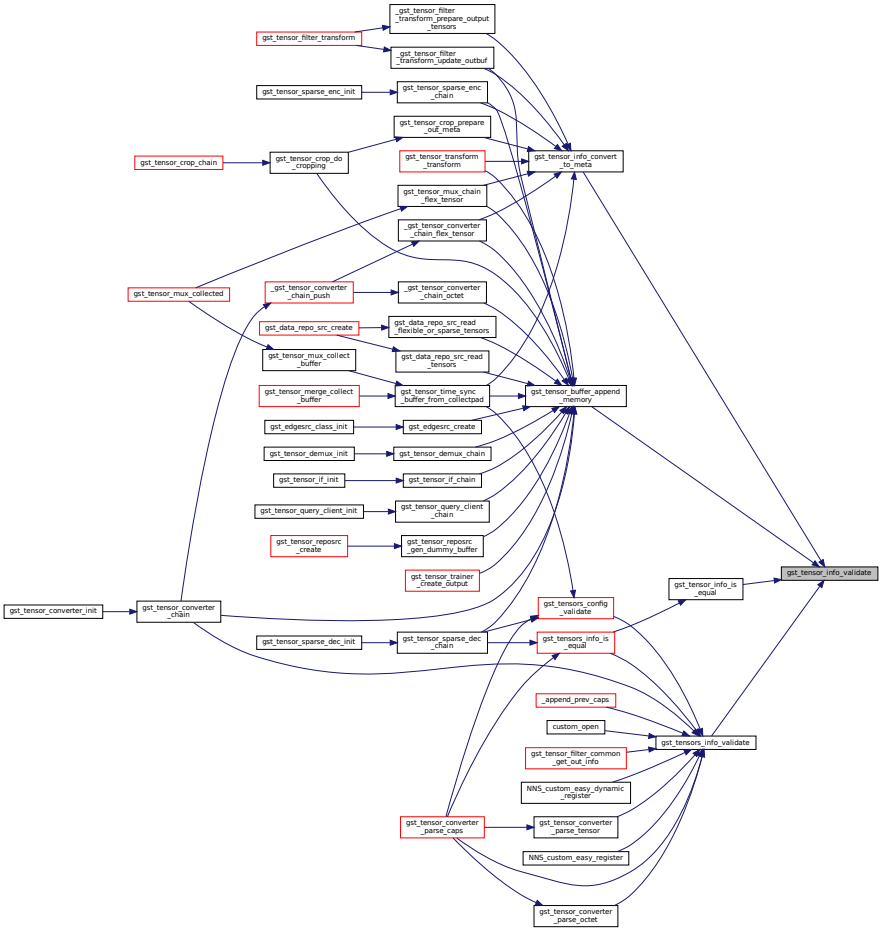
TRUE if info is valid

Definition at line 174 of file nnstreamer_plugin_api_util_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.78.3.24 gst_tensor_meta_info_convert()

```
gboolean gst_tensor_meta_info_convert (
            GstTensorMetaInfo * meta,
            GstTensorInfo * info )
```

Convert GstTensorMetaInfo structure to GstTensorInfo.

Parameters

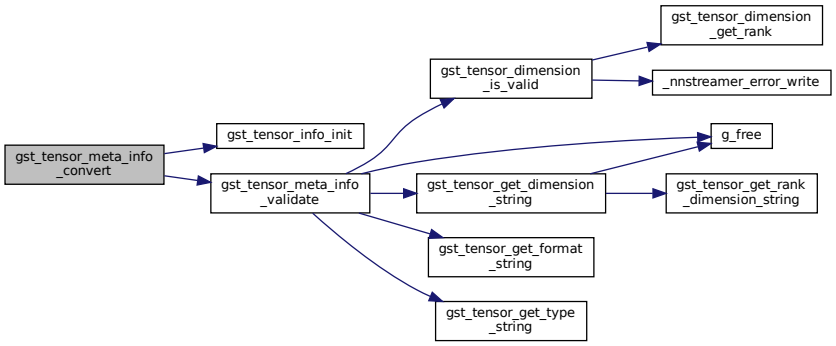
in	*meta*	tensor meta structure to be converted
out	*info*	GstTensorInfo to be filled

Returns

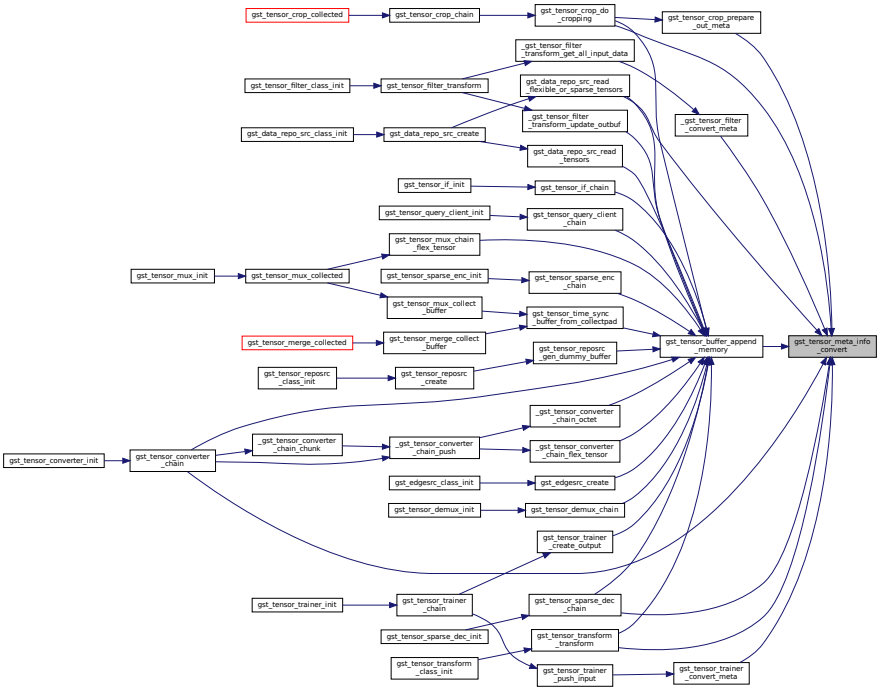
TRUE if successfully set the info

Definition at line 1562 of file nnstreamer_plugin_api_util_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.78.3.25 gst_tensor_meta_info_get_data_size()

```
gsize gst_tensor_meta_info_get_data_size (
            GstTensorMetaInfo * meta )
```

Get the data size calculated from tensor meta.

Parameters

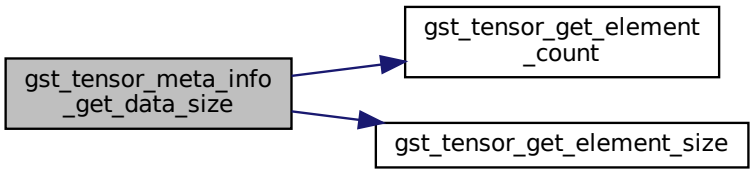
in	*meta*	tensor meta structure

Returns

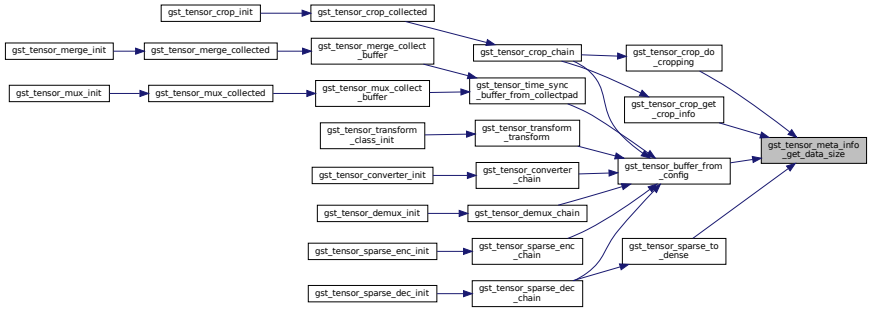
The data size for meta info (0 if meta is invalid)

Definition at line 1477 of file nnstreamer_plugin_api_util_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.78.3.26 gst_tensor_meta_info_get_header_size()

```
gsize gst_tensor_meta_info_get_header_size (
            GstTensorMetaInfo * meta )
```

Get the header size to handle a tensor meta.

Parameters

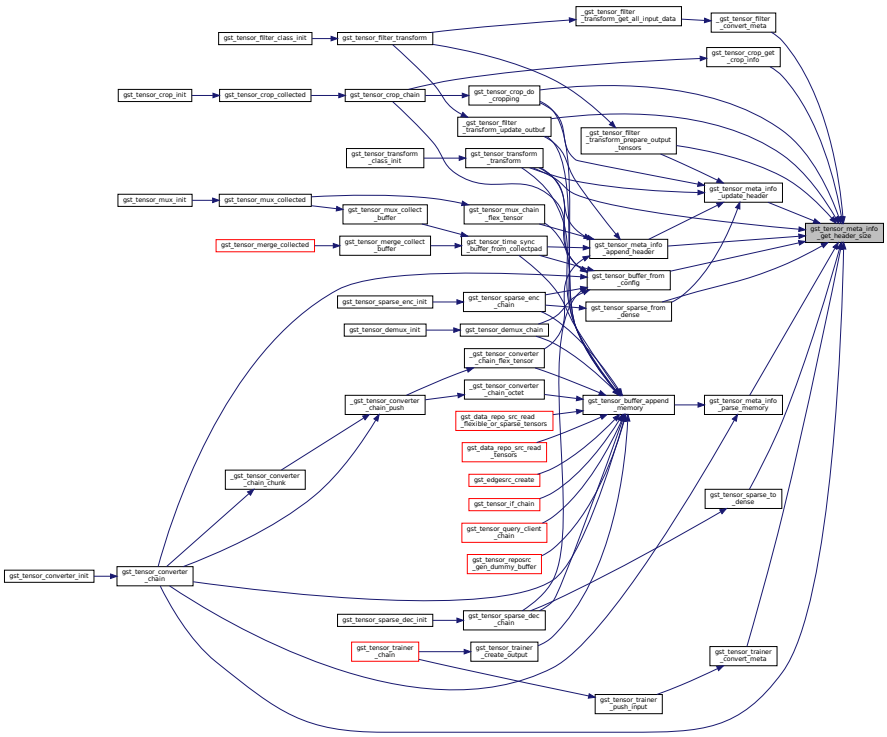
in	*meta*	tensor meta structure

Returns

Header size for meta info (0 if meta is invalid)

Definition at line 1456 of file nnstreamer_plugin_api_util_impl.c.

Here is the caller graph for this function:



9.78.3.27 gst_tensor_meta_info_get_version()

```
void gst_tensor_meta_info_get_version (
            GstTensorMetaInfo * meta,
            guint * major,
            guint * minor )
```

Get the version of tensor meta.

Parameters

in	*meta*	tensor meta structure
out	*major*	pointer to get the major version number
out	*minor*	pointer to get the minor version number

Definition at line 1393 of file nnstreamer_plugin_api_util_impl.c.

9.78.3.28 gst_tensor_meta_info_init()

```
void gst_tensor_meta_info_init (
```

```
            GstTensorMetaInfo * meta )
```

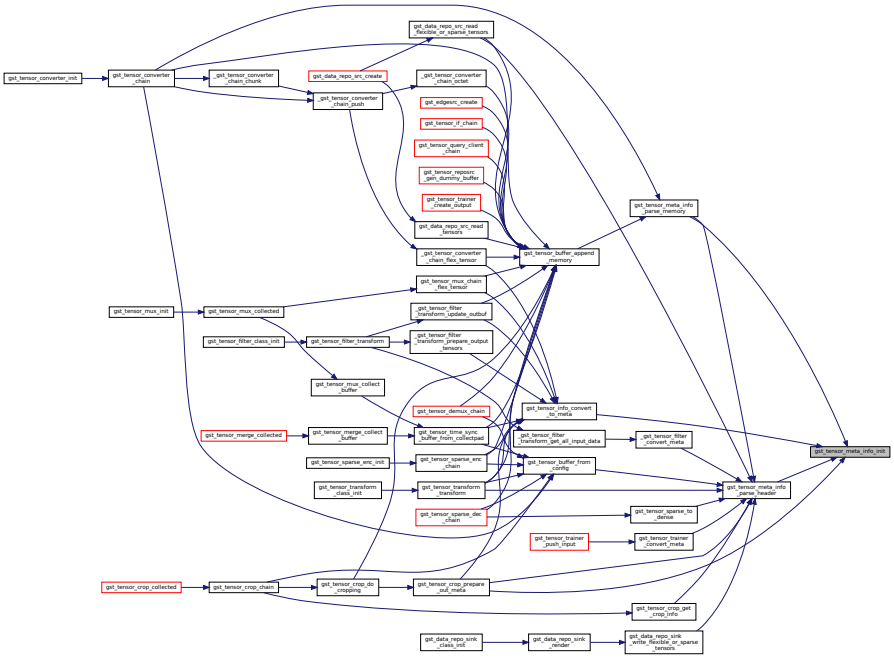
Initialize the tensor meta info structure.

Parameters

in,out	*meta*	tensor meta structure to be initialized

Definition at line 1372 of file nnstreamer_plugin_api_util_impl.c.

Here is the caller graph for this function:



9.78.3.29 gst_tensor_meta_info_parse_header()

```
gboolean gst_tensor_meta_info_parse_header (
            GstTensorMetaInfo * meta,
            gpointer header )
```

Parse header and fill the tensor meta.

Parameters

out	*meta*	tensor meta structure to be filled
in	*header*	pointer to header to be parsed

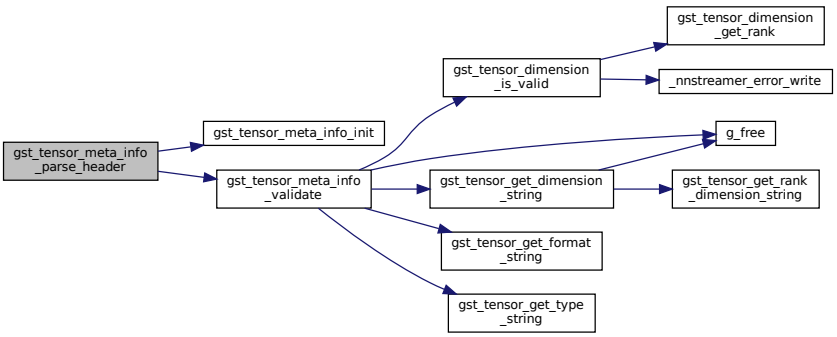
Returns

TRUE if successfully set the meta

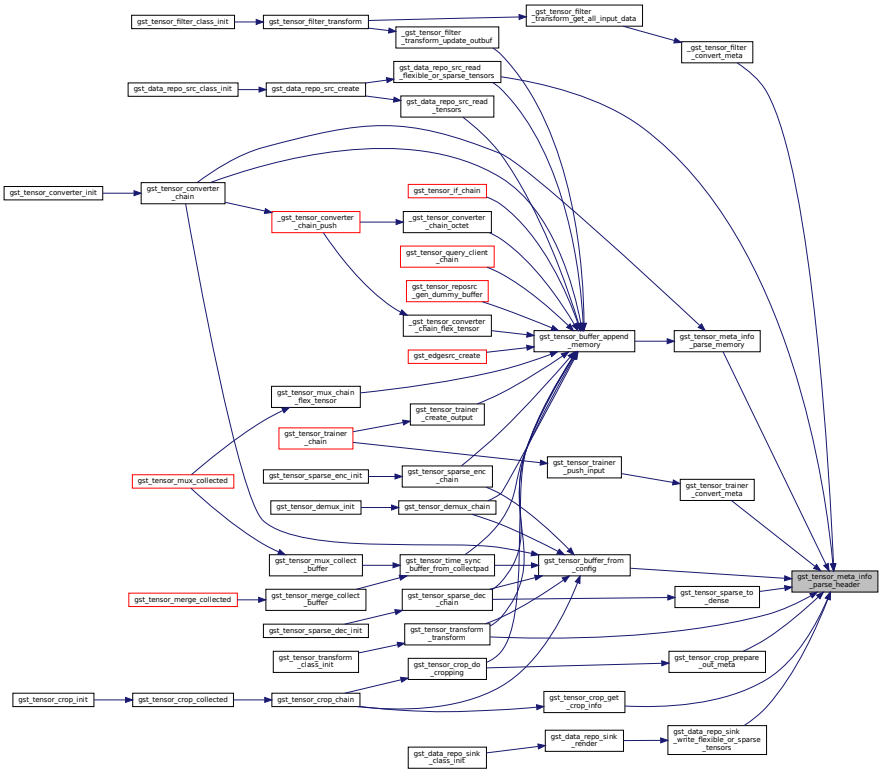
Todo update meta info for each version

Definition at line 1527 of file nnstreamer_plugin_api_util_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.78.3.30 gst_tensor_meta_info_update_header()

```
gboolean gst_tensor_meta_info_update_header (
            GstTensorMetaInfo * meta,
            gpointer header )
```

Update header from tensor meta.

Parameters

in	*meta*	tensor meta structure
out	*header*	pointer to header to be updated

Returns

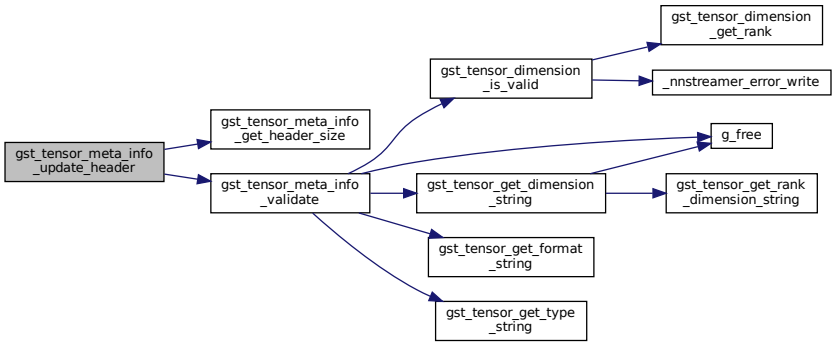
TRUE if successfully set the header

Note

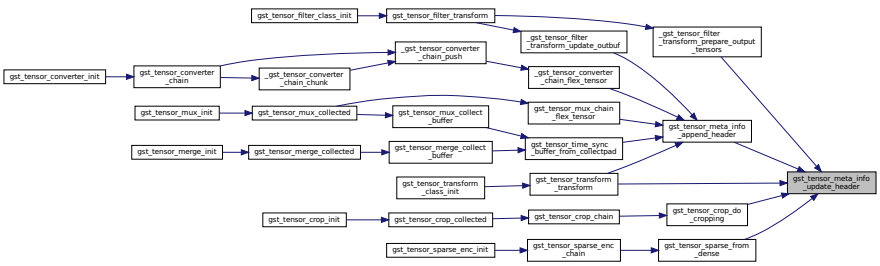
User should allocate enough memory for header (see gst_tensor_meta_info_get_header_size()).

Definition at line 1505 of file nnstreamer_plugin_api_util_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.78.3.31 gst_tensor_meta_info_validate()

```
gboolean gst_tensor_meta_info_validate (
            GstTensorMetaInfo * meta )
```

Check the meta info is valid.

Parameters

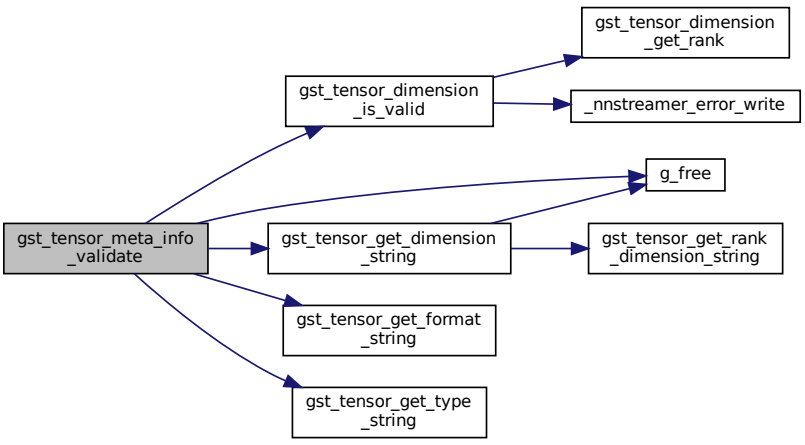
in	*meta*	tensor meta structure

Returns

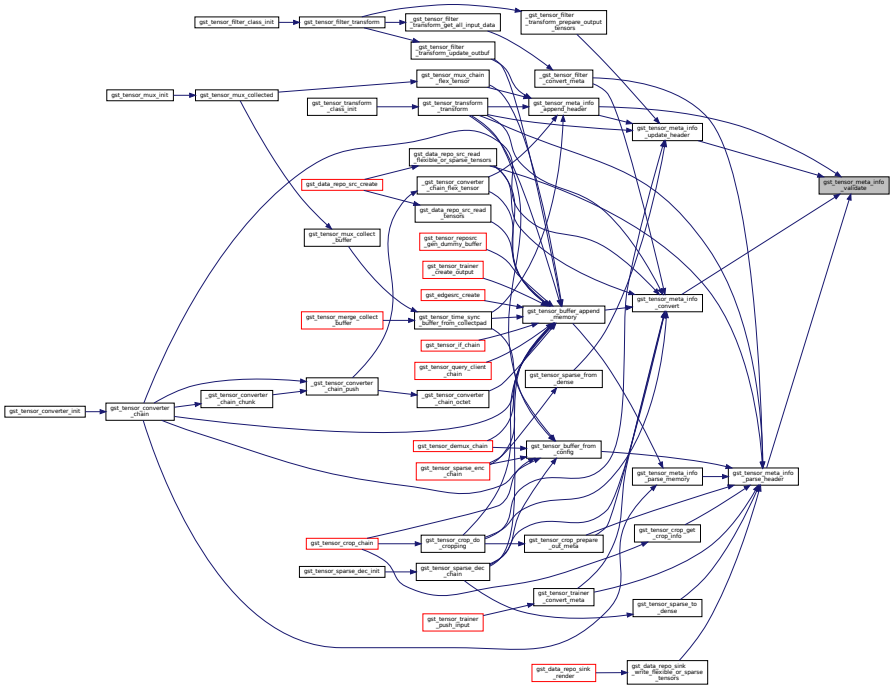
TRUE if given meta is valid

Definition at line 1414 of file nnstreamer_plugin_api_util_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.78.3.32 gst_tensor_parse_dimension()

```
guint gst_tensor_parse_dimension (
            const gchar * dimstr,
            tensor_dim dim )
```

Parse tensor dimension parameter string.

Returns

The Rank. 0 if error.

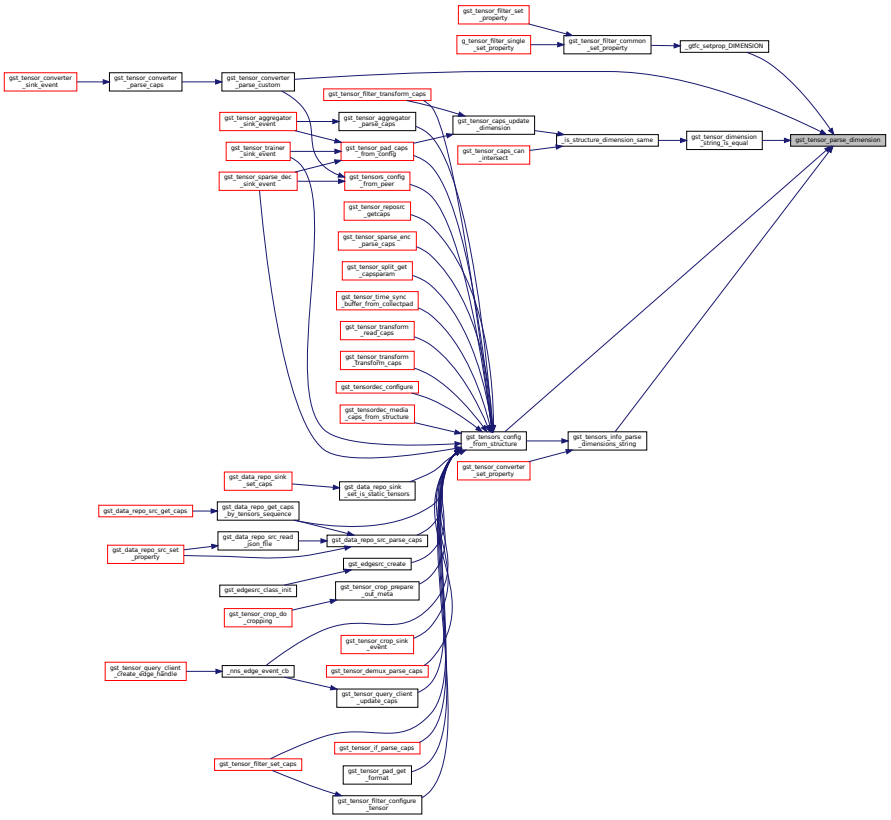
Parameters

dimstr	The dimension string in the format of d1:...:d16, d1:d2:d3, d1:d2, or d1, where dN is a positive integer and d1 is the innermost dimension; i.e., dim[d16]...[d1];
dim	dimension to be filled.

Definition at line 1040 of file nnstreamer_plugin_api_util_impl.c.

Here is the call graph for this function:

Here is the caller graph for this function:



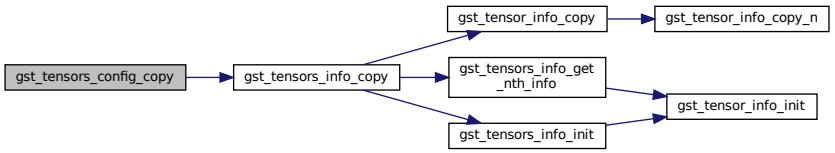
9.78.3.33 gst_tensors_config_copy()

```
void gst_tensors_config_copy (
            GstTensorsConfig * dest,
            const GstTensorsConfig * src )
```

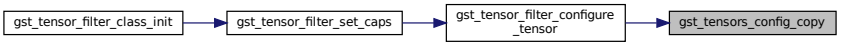
Copy tensors config.

Definition at line 904 of file nnstreamer_plugin_api_util_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.78.3.34 gst_tensors_config_free()

```
void gst_tensors_config_free (
            GstTensorsConfig * config )
```

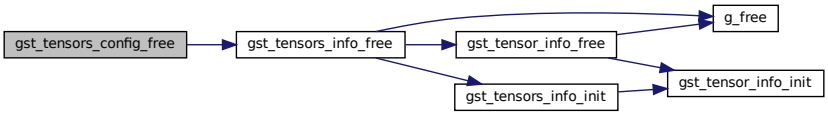
Free allocated data in tensors config structure.

Parameters

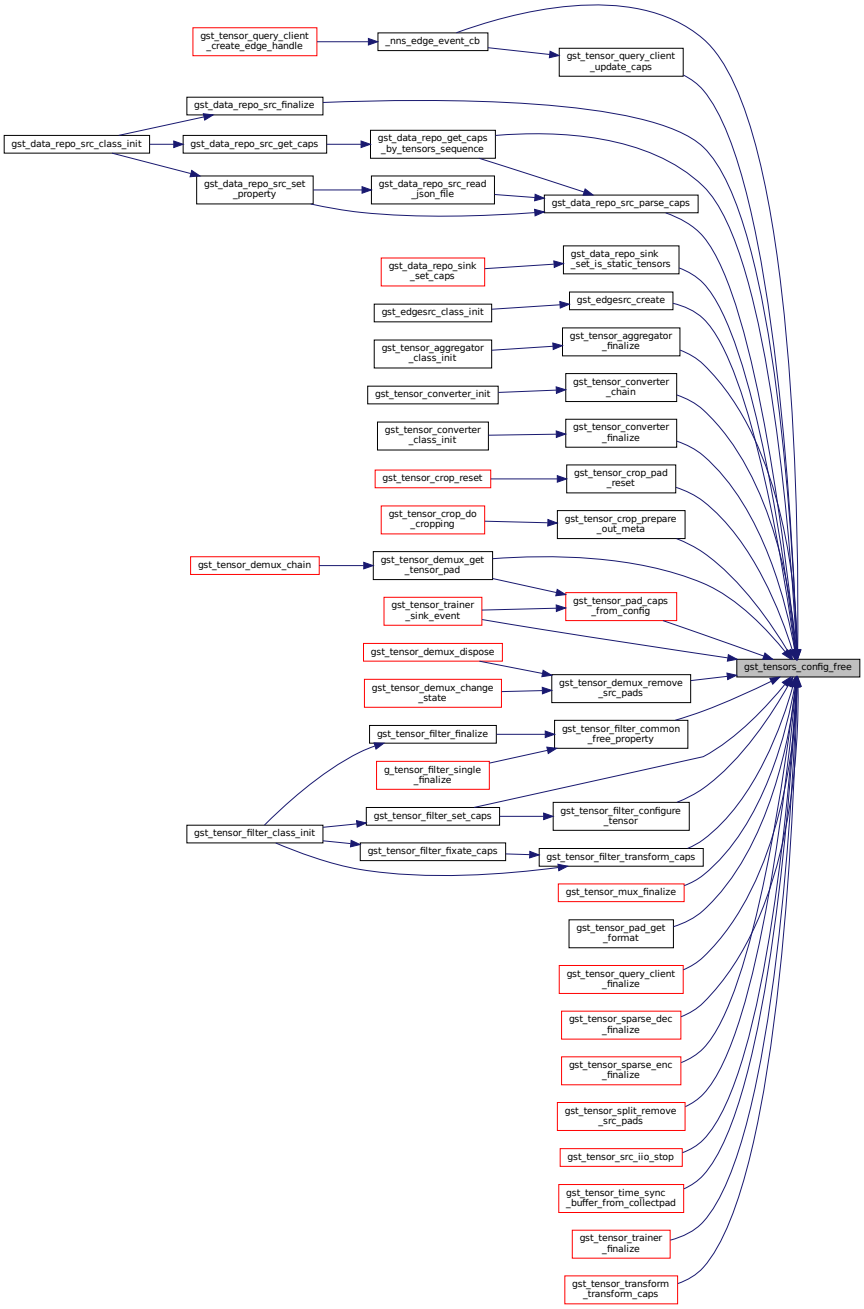
config	tensors config structure

Definition at line 845 of file nnstreamer_plugin_api_util_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.78.3.35 gst_tensors_config_init()

```
void gst_tensors_config_init (
            GstTensorsConfig * config )
```

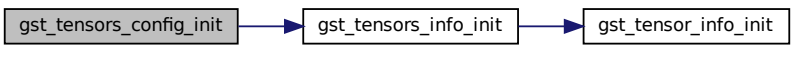
Initialize the tensors config info structure (for other/tensors)

Parameters

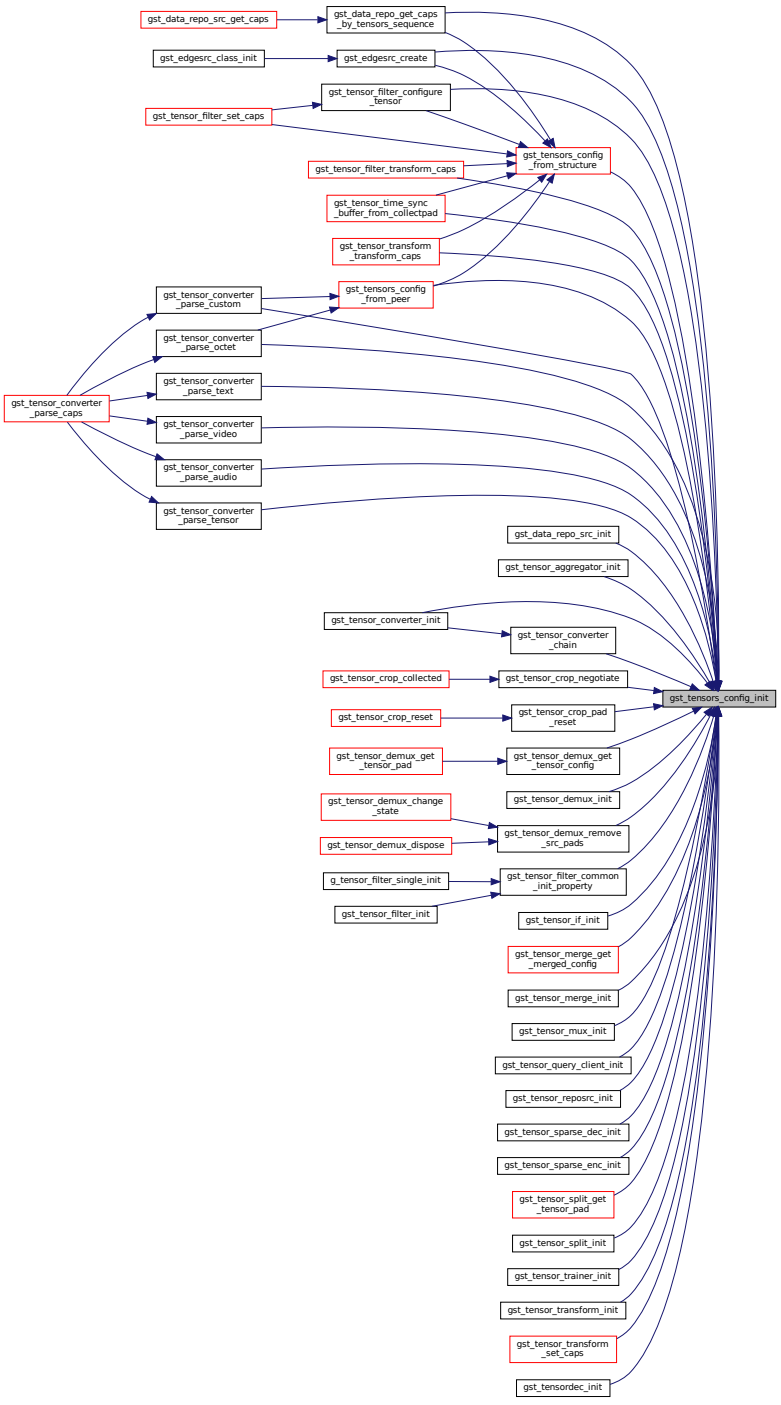
config	tensors config structure to be initialized

Definition at line 830 of file nnstreamer_plugin_api_util_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.78.3.36 gst_tensors_config_is_equal()

```
gboolean gst_tensors_config_is_equal (
            const GstTensorsConfig * c1,
            const GstTensorsConfig * c2 )
```

Compare tensor config info (for other/tensors)

Parameters

TRUE	if equal

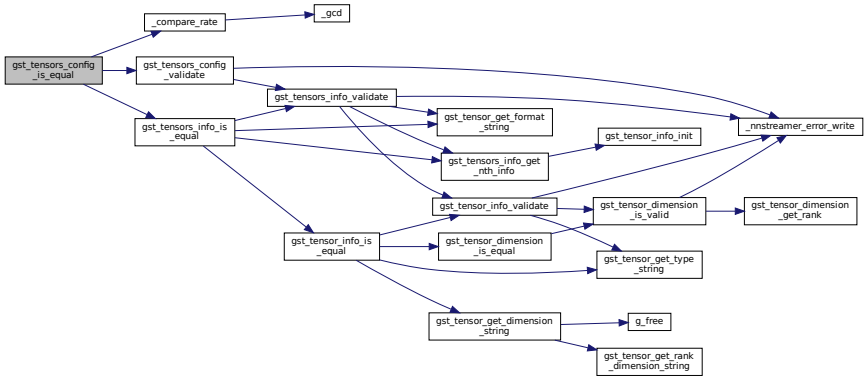
Compare tensor config info (for other/tensors)

Parameters

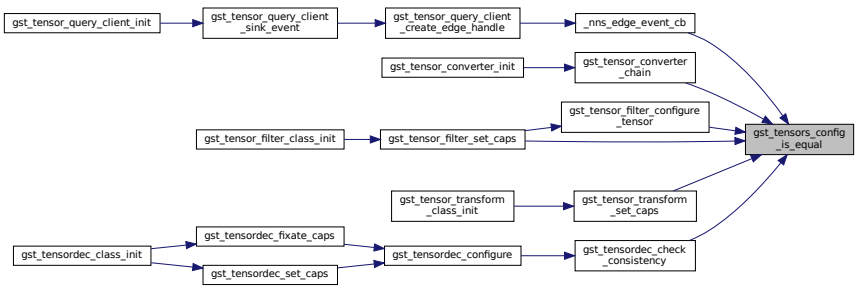
TRUE	if equal

Definition at line 881 of file nnstreamer_plugin_api_util_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.78.3.37 gst_tensors_config_to_string()

```
gchar* gst_tensors_config_to_string (
            const GstTensorsConfig * config )
```

Tensor config represented as a string. Caller should free it.

Parameters

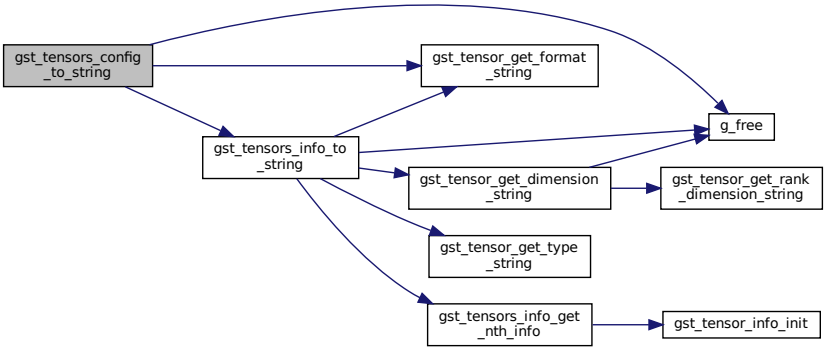
config	tensor config structure

Returns

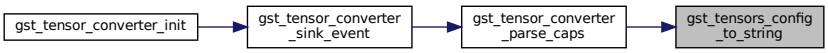
The newly allocated string representing the config. Free after use.

Definition at line 920 of file nnstreamer_plugin_api_util_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.78.3.38 gst_tensors_config_validate()

```
gboolean gst_tensors_config_validate (
            const GstTensorsConfig * config )
```

Check the tensors are all configured (for other/tensors)

Parameters

config	tensor config structure

Returns

TRUE if configured

Check the tensors are all configured (for other/tensors)

Parameters

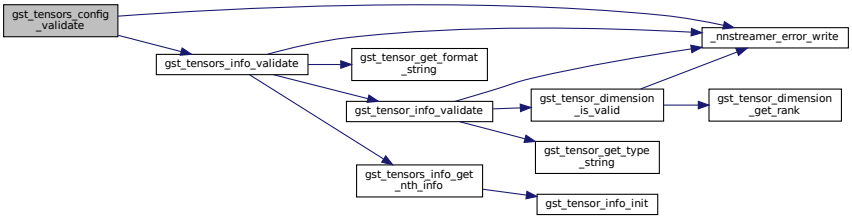
config	tensor config structure

Returns

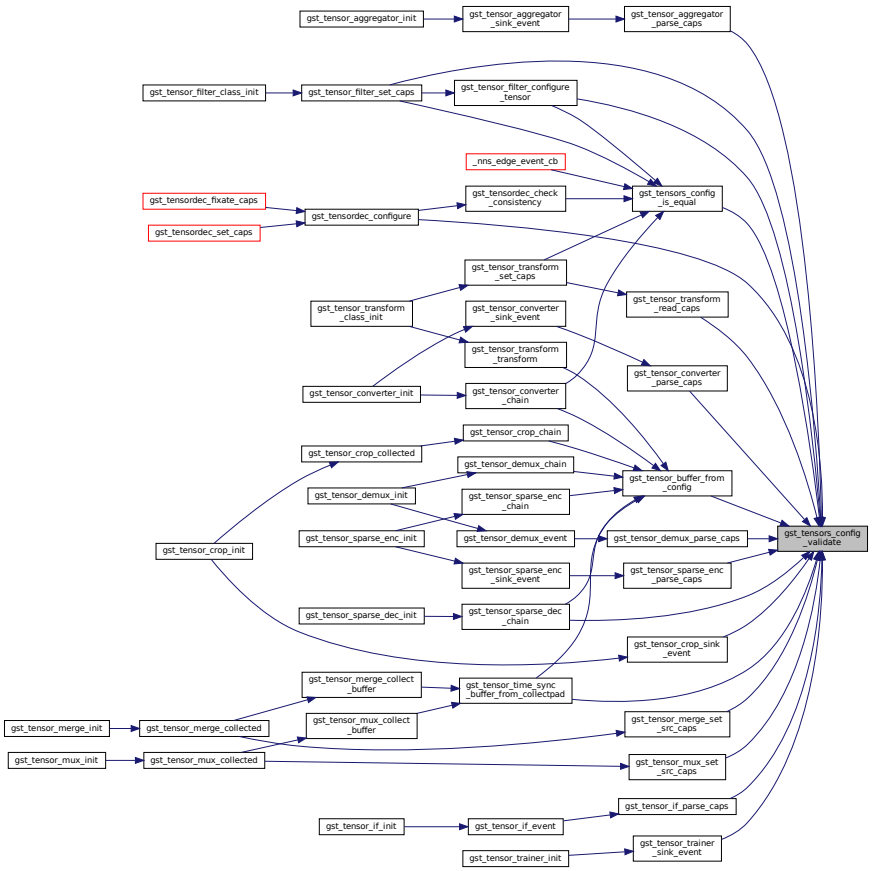
TRUE if configured

Definition at line 858 of file nnstreamer_plugin_api_util_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.78.3.39 gst_tensors_info_copy()

```
void gst_tensors_info_copy (
            GstTensorsInfo * dest,
            const GstTensorsInfo * src )
```

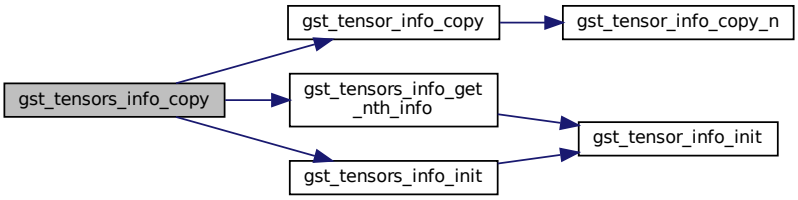
Copy tensor info.

Note

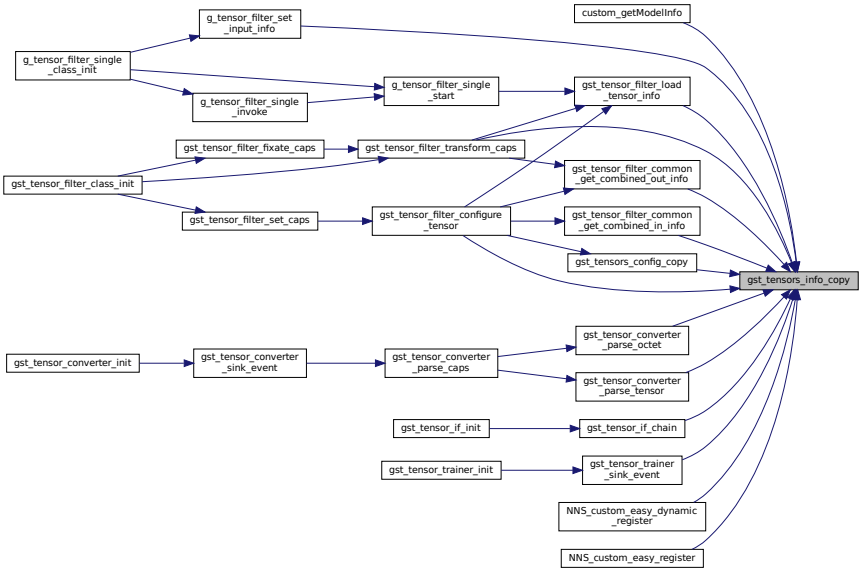
Copied info should be freed with gst_tensors_info_free()

Definition at line 502 of file nnstreamer_plugin_api_util_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.78.3.40 gst_tensors_info_free()

```
void gst_tensors_info_free (
            GstTensorsInfo * info )
```

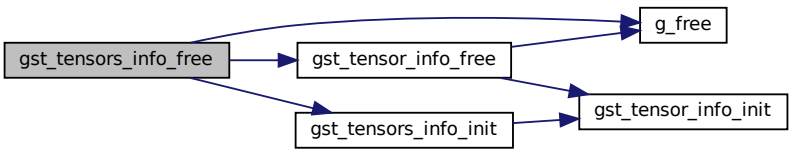
Free allocated data in tensors info structure.

Parameters

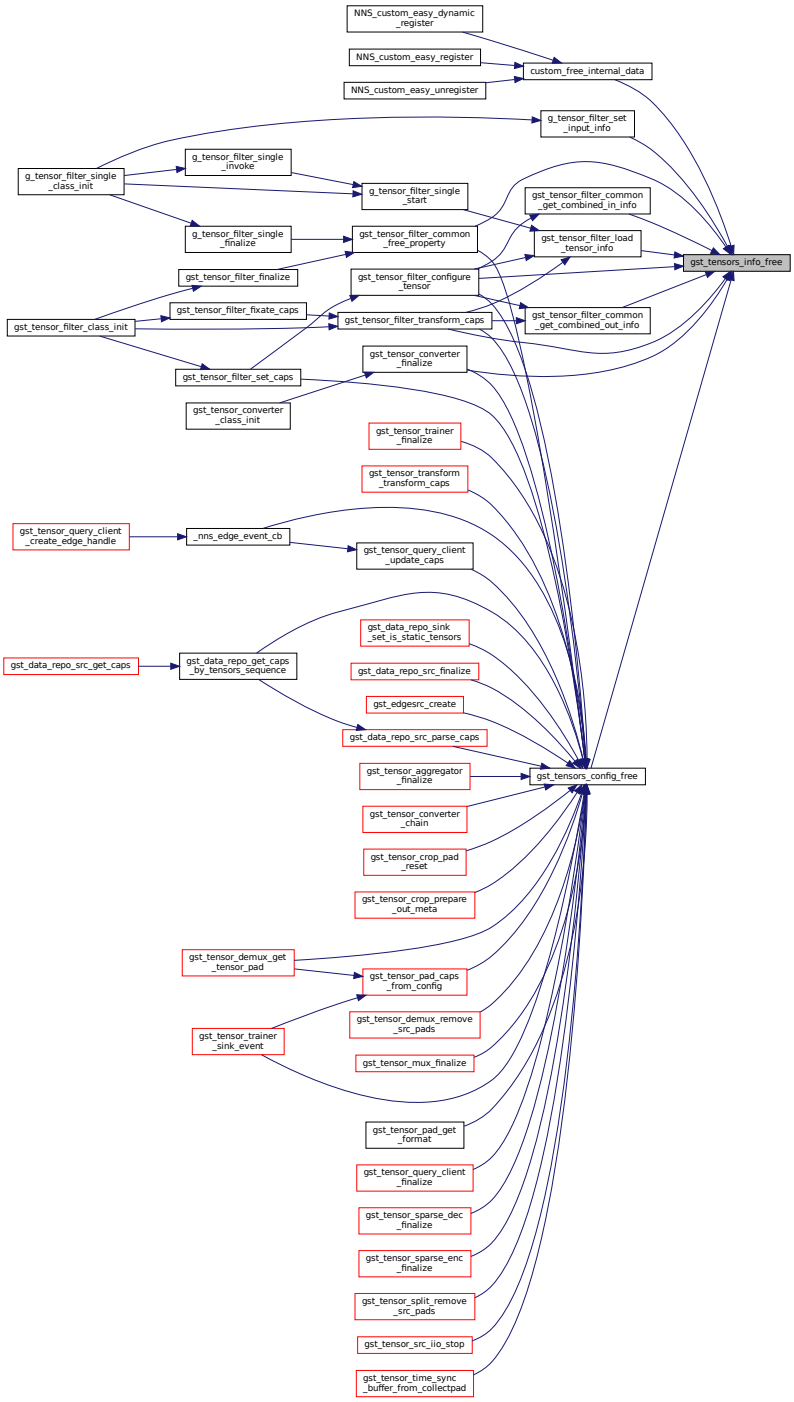
info	tensors info structure

Definition at line 347 of file nnstreamer_plugin_api_util_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.78.3.41 gst_tensors_info_get_dimensions_string()

```
gchar* gst_tensors_info_get_dimensions_string (
            const GstTensorsInfo * info )
```

Get the string of dimensions in tensors info.

Parameters

info	tensors info structure

Returns

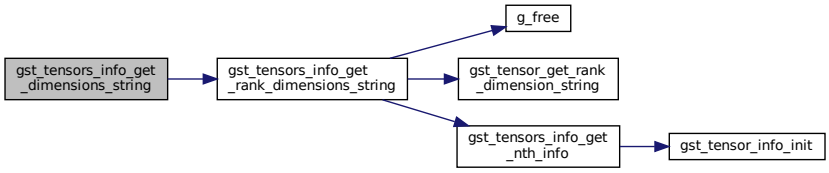
string of dimensions in tensors info (NULL if the number of tensors is 0)

Note

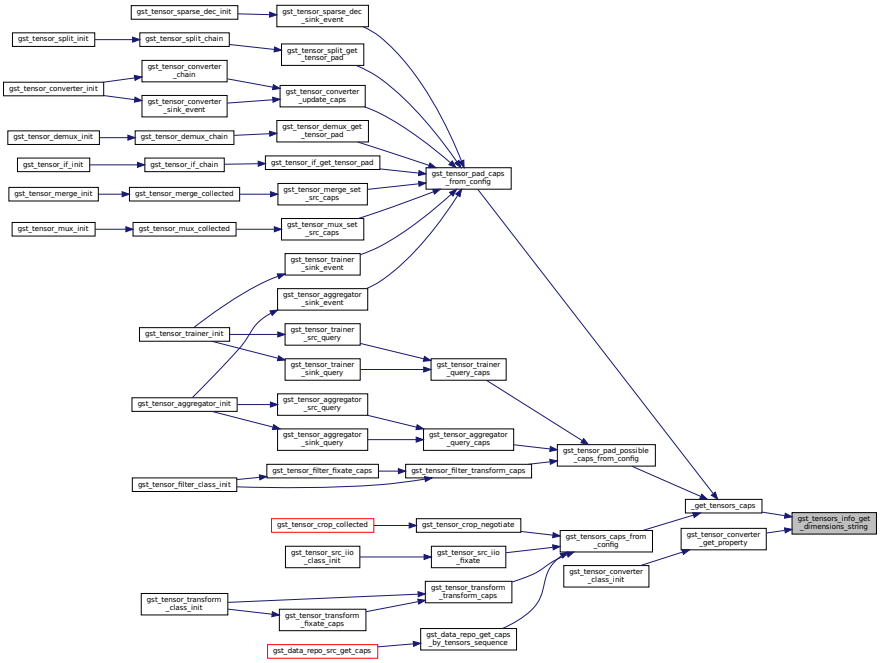
The returned value should be freed with g_free()

Definition at line 661 of file nnstreamer_plugin_api_util_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.78.3.42 gst_tensors_info_get_names_string()

```
gchar* gst_tensors_info_get_names_string (
            const GstTensorsInfo * info )
```

Get the string of tensor names in tensors info.

Parameters

info	tensors info structure

Returns

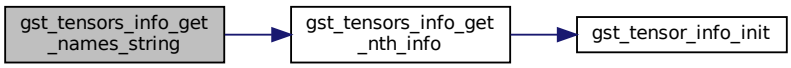
string of names in tensors info (NULL if the number of tensors is 0)

Note

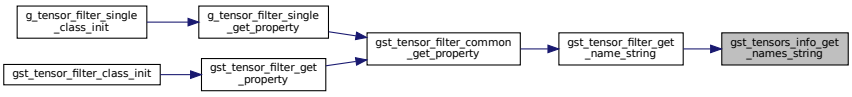
The returned value should be freed with g_free()

Definition at line 749 of file nnstreamer_plugin_api_util_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.78.3.43 gst_tensors_info_get_nth_info()

```
GstTensorInfo* gst_tensors_info_get_nth_info (
            GstTensorsInfo * info,
            guint index )
```

Get the pointer of nth tensor information.

Parameters

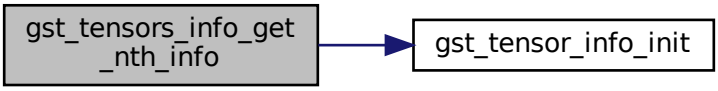
info	tensors info structure
index	the index of tensor to be fetched

Returns

The pointer to tensor info structure

Definition at line 296 of file nnstreamer_plugin_api_util_impl.c.

Here is the call graph for this function:



9.78.3.44 gst_tensors_info_get_rank_dimensions_string()

```
gchar* gst_tensors_info_get_rank_dimensions_string (
            const GstTensorsInfo * info,
            const unsigned int rank )
```

Get the string of dimensions in tensors info and rank count.

Parameters

info	tensors info structure
rank	rank count of given tensor dimension

Returns

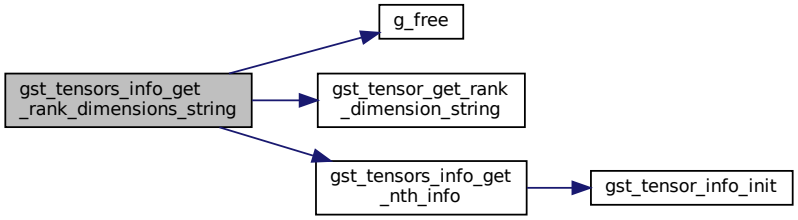
Formatted string of given dimension

Note

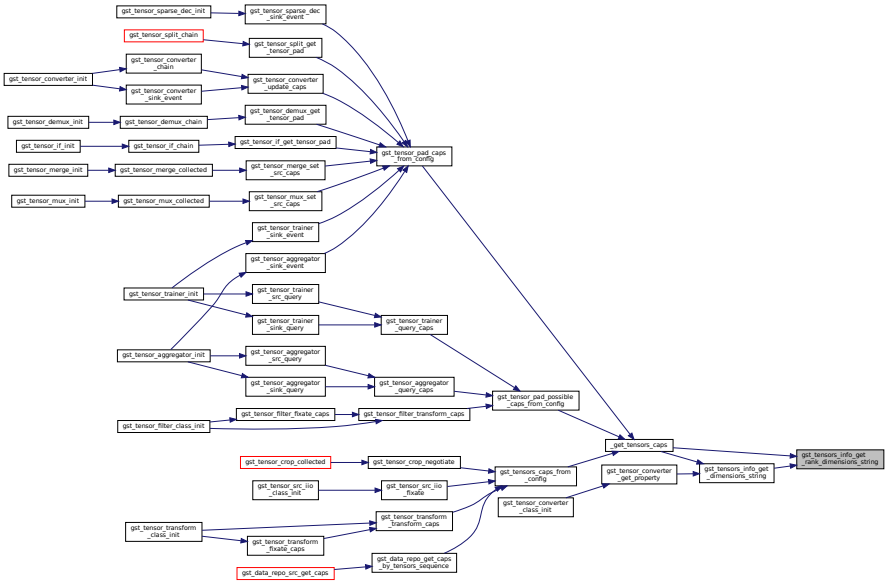
If rank count is 3, then returned string is 'd1:d2:d3`. The returned value should be freed with g_free()

Definition at line 676 of file nnstreamer_plugin_api_util_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.78.3.45 gst_tensors_info_get_size()

```
gsize gst_tensors_info_get_size (
            const GstTensorsInfo * info,
            gint index )
```

Get data size of single tensor.

Parameters

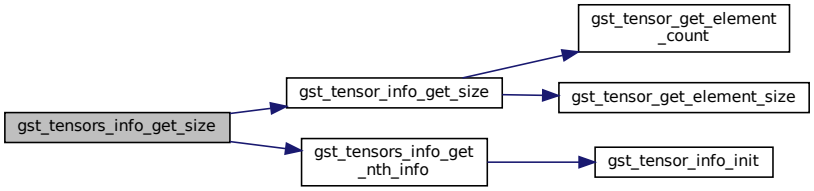
info	tensors info structure
index	the index of tensor (-1 to get total size of tensors)

Returns

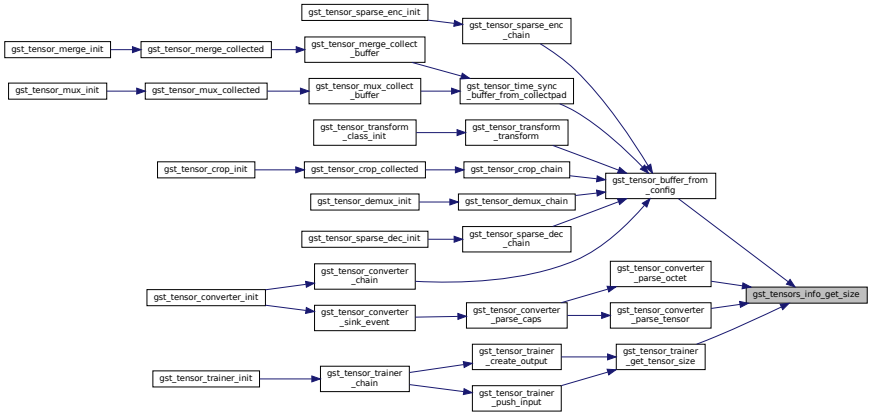
data size

Definition at line 376 of file nnstreamer_plugin_api_util_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.78.3.46 gst_tensors_info_get_types_string()

```
gchar* gst_tensors_info_get_types_string (
            const GstTensorsInfo * info )
```

Get the string of types in tensors info.

Parameters

info	tensors info structure

Returns

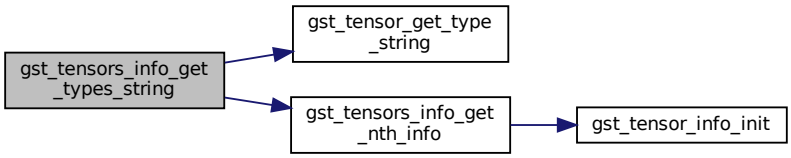
string of types in tensors info (NULL if the number of tensors is 0)

Note

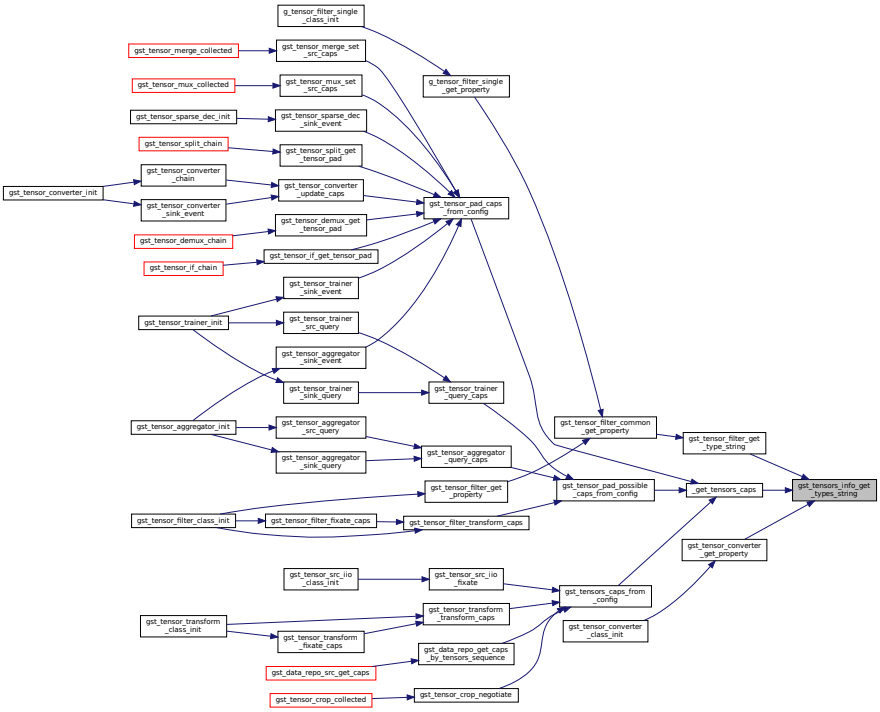
The returned value should be freed with g_free()

Definition at line 714 of file nnstreamer_plugin_api_util_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.78.3.47 gst_tensors_info_init()

```
void gst_tensors_info_init (
            GstTensorsInfo * info )
```

Initialize the tensors info structure.

Parameters

info	tensors info structure to be initialized

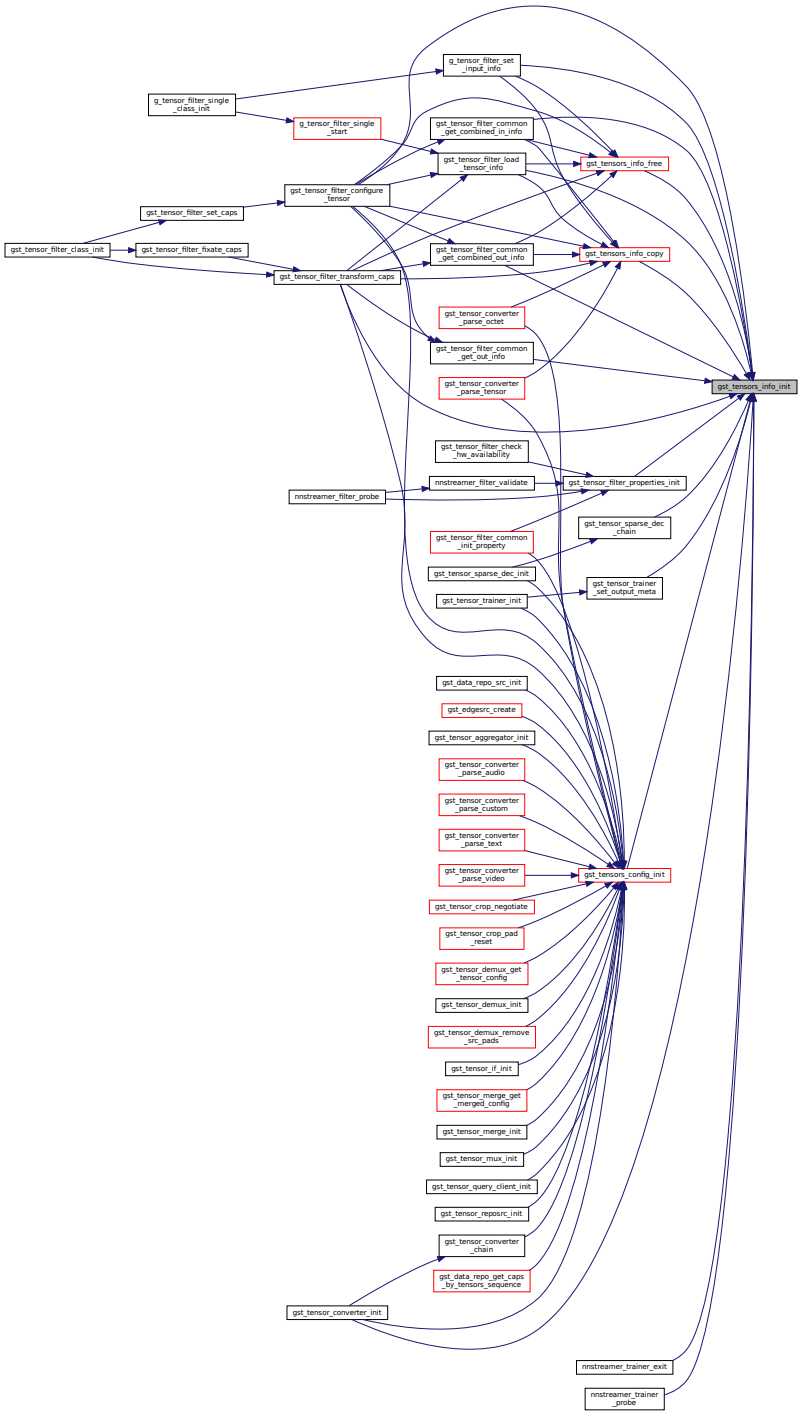
Note

default format is static

Definition at line 325 of file nnstreamer_plugin_api_util_impl.c.

Here is the call graph for this function:

Here is the caller graph for this function:



9.78.3.48 gst_tensors_info_is_equal()

```
gboolean gst_tensors_info_is_equal (
            const GstTensorsInfo * i1,
            const GstTensorsInfo * i2 )
```

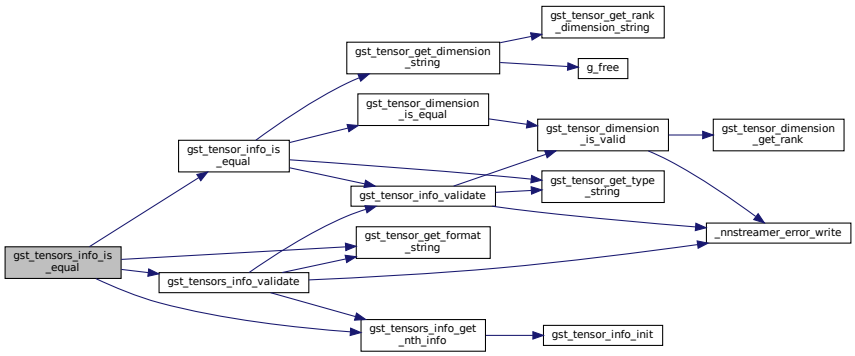
Compare tensors info.

Returns

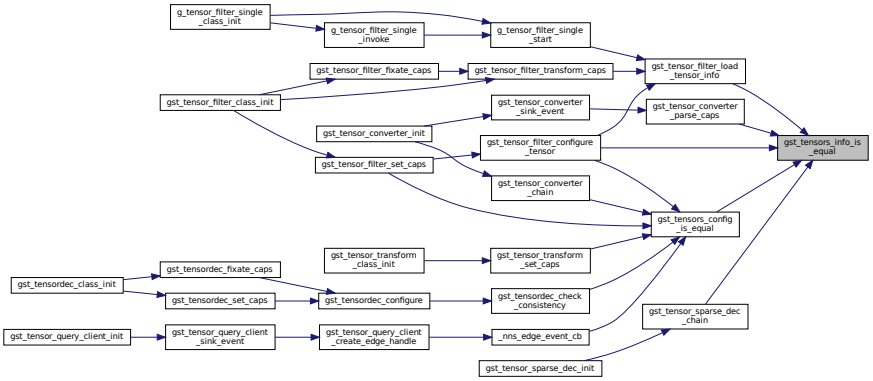
TRUE if equal, FALSE if given tensor infos are invalid or not equal.

Definition at line 454 of file nnstreamer_plugin_api_util_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.78.3.49 gst_tensors_info_parse_dimensions_string()

```
guint gst_tensors_info_parse_dimensions_string (
            GstTensorsInfo * info,
            const gchar * dim_string )
```

Parse the string of dimensions.

Parameters

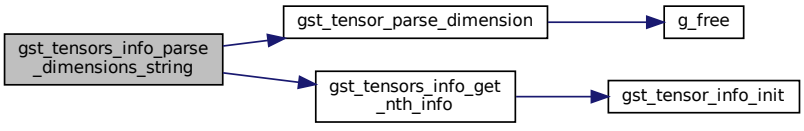
info	tensors info structure
dim_string	string of dimensions

Returns

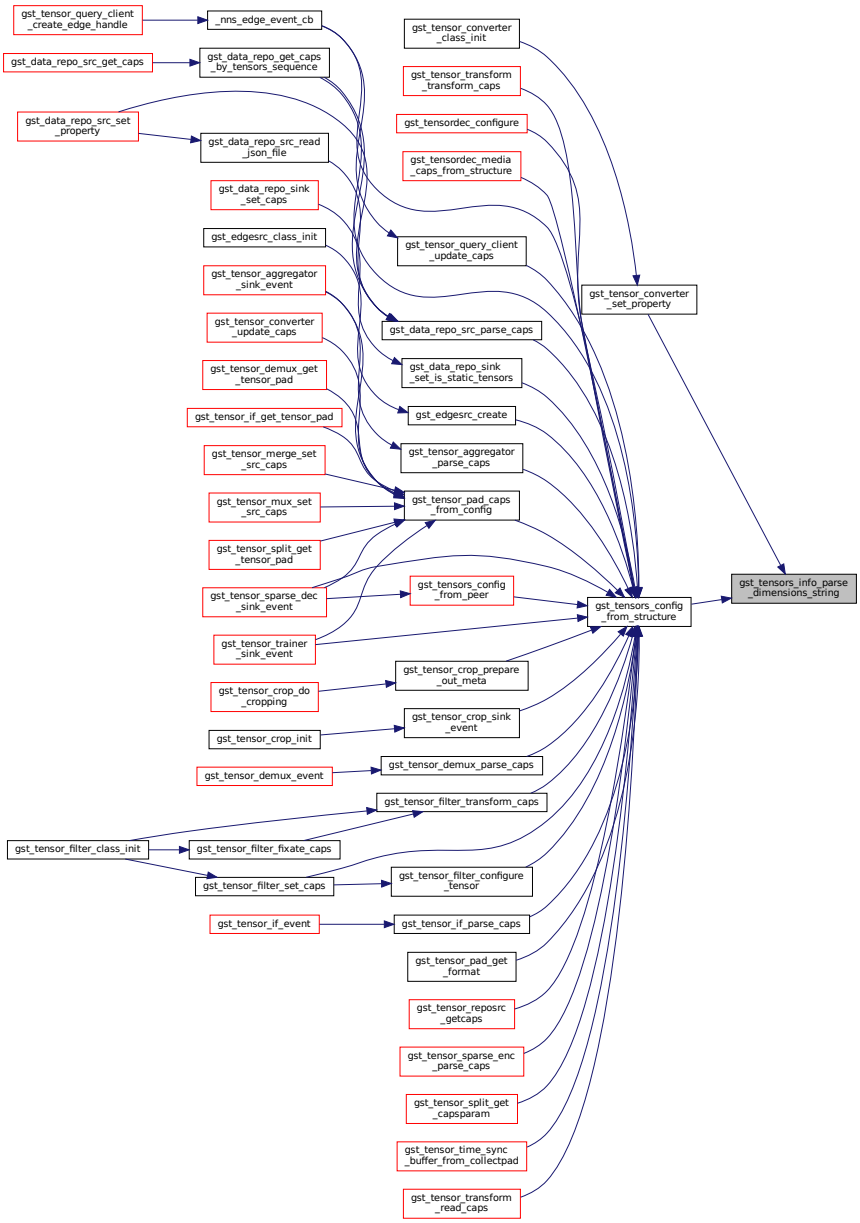
number of parsed dimensions

Definition at line 532 of file nnstreamer_plugin_api_util_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.78.3.50 gst_tensors_info_parse_names_string()

```
guint gst_tensors_info_parse_names_string (
            GstTensorsInfo * info,
            const gchar * name_string )
```

Parse the string of names.

Parameters

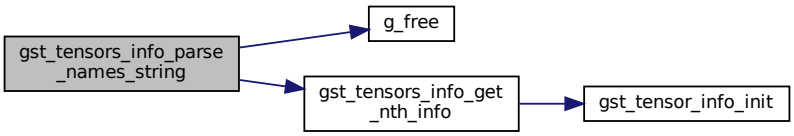
info	tensors info structure
name_string	string of names

Returns

number of parsed names

Definition at line 612 of file nnstreamer_plugin_api_util_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:

9.78.3.51 gst_tensors_info_parse_types_string()

```
guint gst_tensors_info_parse_types_string (
            GstTensorsInfo * info,
            const gchar * type_string )
```

Parse the string of types.

Parameters

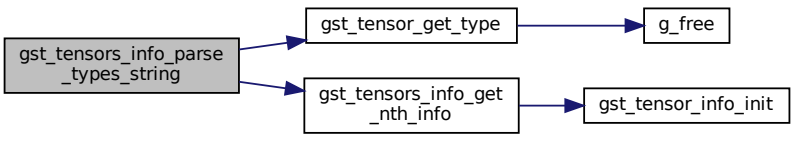
info	tensors info structure
type_string	string of types

Returns

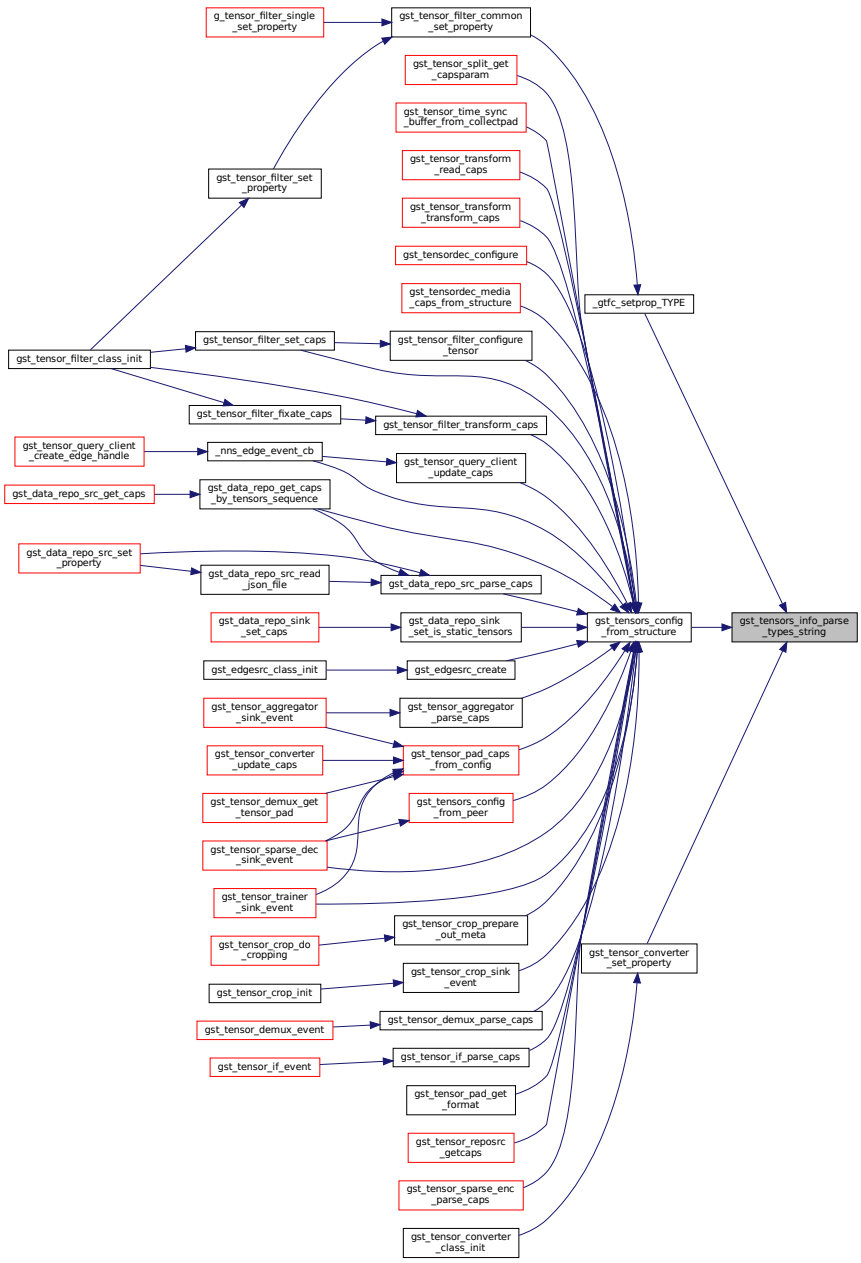
number of parsed types

Definition at line 572 of file nnstreamer_plugin_api_util_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.78.3.52 gst_tensors_info_to_string()

```
gchar* gst_tensors_info_to_string (
            const GstTensorsInfo * info )
```

GstTensorsInfo represented as a string. Caller should free it.

Parameters

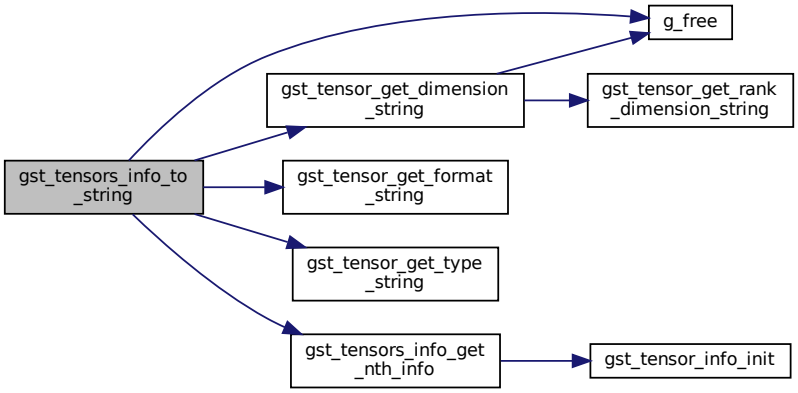
info	GstTensorsInfo structure

Returns

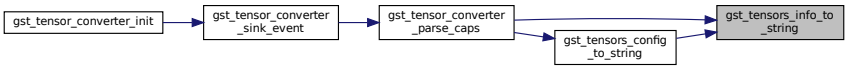
The newly allocated string representing the tensors info. Free after use.

Definition at line 783 of file nnstreamer_plugin_api_util_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.78.3.53 gst_tensors_info_validate()

```
gboolean gst_tensors_info_validate (
            const GstTensorsInfo * info )
```

Check the tensors info is valid.

Parameters

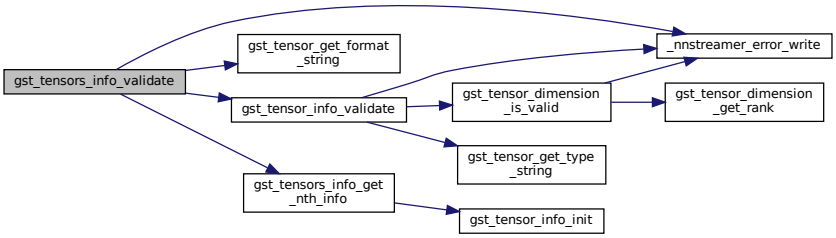
info	tensors info structure

Returns

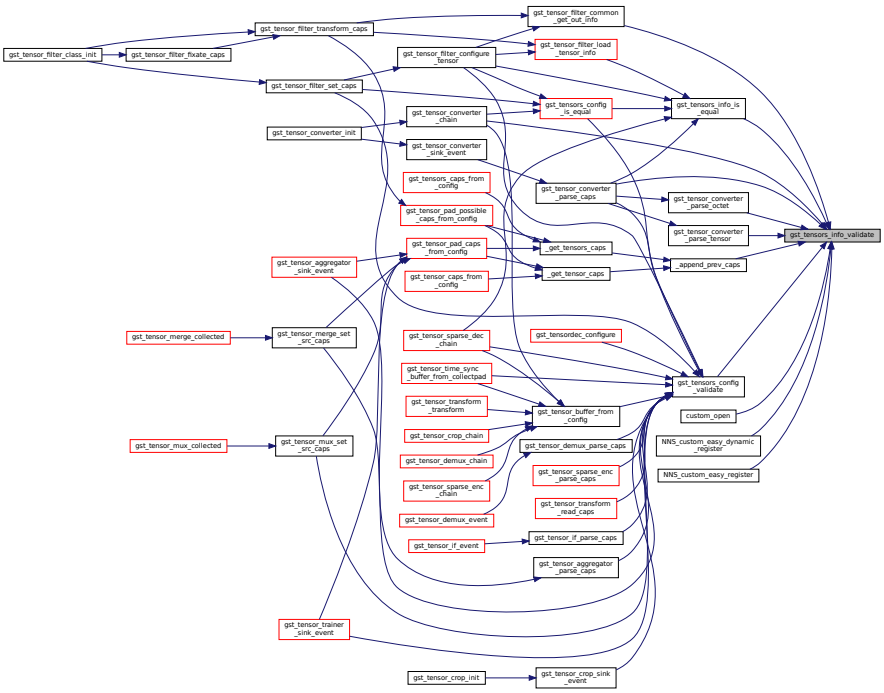
TRUE if info is valid

Definition at line 404 of file nnstreamer_plugin_api_util_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.78.3.54 nnstreamer_version_fetch()

```
void nnstreamer_version_fetch (
            guint * major,
            guint * minor,
            guint * micro )
```

Get the version of NNStreamer (int, divided).

Parameters

out	*major*	MAJOR.minor.micro, won't set if it's null.
out	*minor*	major.MINOR.micro, won't set if it's null.
out	*micro*	major.minor.MICRO, won't set if it's null.

Definition at line 1624 of file nnstreamer_plugin_api_util_impl.c.

9.78.3.55 nnstreamer_version_string()

```
gchar* nnstreamer_version_string (
            void  )
```

Get the version of NNStreamer.

Returns

Newly allocated string. The returned string should be freed with g_free().

Get the version of NNStreamer.

Returns

Newly allocated string. The returned string should be freed with g_free().

Definition at line 1609 of file nnstreamer_plugin_api_util_impl.c.

9.79 nnstreamer/include/nnstreamer_util.h File Reference

Optional NNStreamer utility functions for sub-plugin writers and users.

This graph shows which files directly or indirectly include this file:

Macros

- #define UNUSED(expr) do { (void)(expr); } while (0)

  Utility to silence unused parameter warning for intentionally unused parameters (e.g., callback functions of a framework)
- #define _g_memdup(data, size) g_memdup (data, size)

  g_memdup() function replaced by g_memdup2() in glib version >= 2.68

9.79.1 Detailed Description

Optional NNStreamer utility functions for sub-plugin writers and users.

NNStreamer Common Utility Header for Plug-Ins and Users Copyright (C) 2021 MyungJoo Ham myungjoo.ham@samsung.com

Date

28 Jul 2021

See also

https://github.com/nnstreamer/nnstreamer

Author

MyungJoo Ham myungjoo.ham@samsung.com

Bug No known bugs except for NYI items

9.79.2 Macro Definition Documentation

9.79.2.1 _g_memdup

```
#define _g_memdup(
              data,
              size ) g_memdup (data, size)
```

g_memdup() function replaced by g_memdup2() in glib version >= 2.68

Definition at line 31 of file nnstreamer_util.h.

9.79.2.2 UNUSED

```
#define UNUSED(
              expr ) do { (void)(expr); } while (0)
```

Utility to silence unused parameter warning for intentionally unused parameters (e.g., callback functions of a framework)

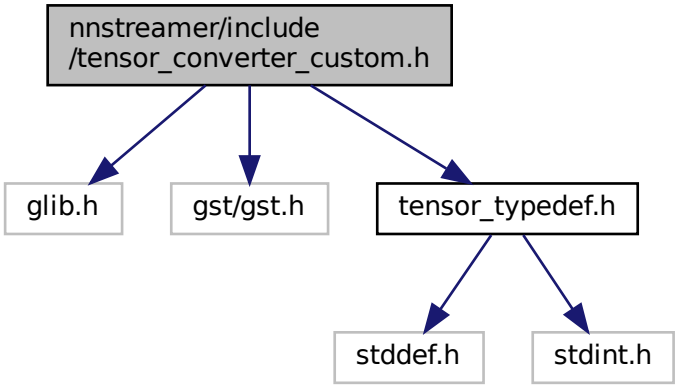
Definition at line 22 of file nnstreamer_util.h.

9.80 nnstreamer/include/tensor_converter_custom.h File Reference

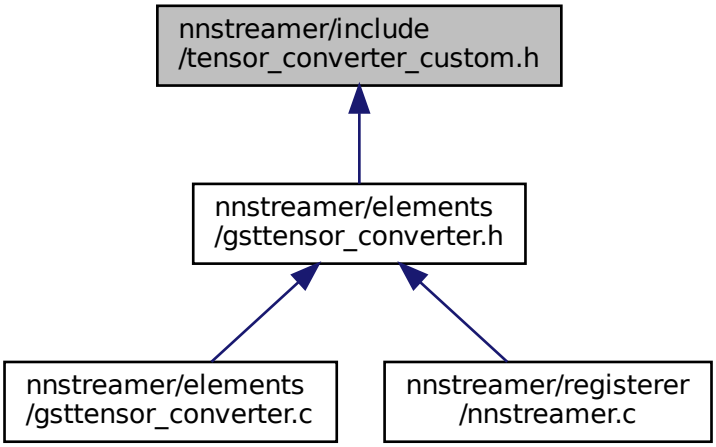
NNStreamer APIs for tensor_converter custom condition.

```
#include <glib.h>
#include <gst/gst.h>
#include "tensor_typedef.h"
```

Include dependency graph for tensor_converter_custom.h:



This graph shows which files directly or indirectly include this file:



Functions

- int nnstreamer_converter_custom_register (const gchar *name, tensor_converter_custom func, void *data)

  Register the custom callback function.
- int nnstreamer_converter_custom_unregister (const gchar *name)

  Unregister the custom callback function.

Variables

- G_BEGIN_DECLS typedef GstBuffer *(* tensor_converter_custom)(GstBuffer *in_buf, void *data, GstTensorsConfig *config)

  Convert to tensors as customized operation.

9.80.1 Detailed Description

NNStreamer APIs for tensor_converter custom condition.

GStreamer/NNStreamer Tensor-Converter Copyright (C) 2021 Gichan Jang gichan2.jang@samsung.com

Date

18 Mar 2021

See also

https://github.com/nnstreamer/nnstreamer

Author

Gichan Jang gichan2.jang@samsung.com

Bug No known bugs except for NYI items

9.80.2 Function Documentation

9.80.2.1 nnstreamer_converter_custom_register()

```
int nnstreamer_converter_custom_register (
            const gchar * name,
            tensor_converter_custom func,
            void * data )
```

Register the custom callback function.

Parameters

in	*name*	The name of tensor_converter custom callback function.
in	*func*	The custom condition function body
	[in/out]	data The internal data for the function

Returns

0 if success. -ERRNO if error.

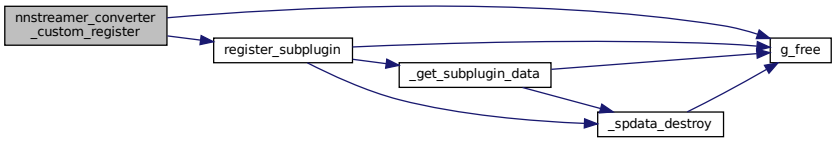
Register the custom callback function.

Returns

0 if success. -ERRNO if error.

Definition at line 2473 of file gsttensor_converter.c.

Here is the call graph for this function:



9.80.2.2 nnstreamer_converter_custom_unregister()

```
int nnstreamer_converter_custom_unregister (
            const gchar * name )
```

Unregister the custom callback function.

Parameters

in	*name*	The registered name of tensor_converter custom callback function.

Returns

0 if success. -ERRNO if error.

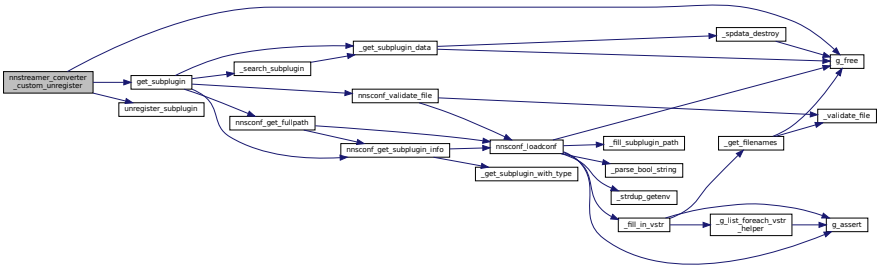
Unregister the custom callback function.

Returns

0 if success. -ERRNO if error.

Definition at line 2500 of file gsttensor_converter.c.

Here is the call graph for this function:



9.80.3 Variable Documentation

9.80.3.1 tensor_converter_custom

```
G_BEGIN_DECLS typedef GstBuffer*(* tensor_converter_custom) (GstBuffer *in_buf, void *data,
GstTensorsConfig *config)
```

Convert to tensors as customized operation.

Parameters

in	*in_buf*	the input stream buffer
	[in/out]	data private data for the callback
out	*config*	tensors config structure to be filled

Returns

output buffer filled by user

Definition at line 31 of file tensor_converter_custom.h.

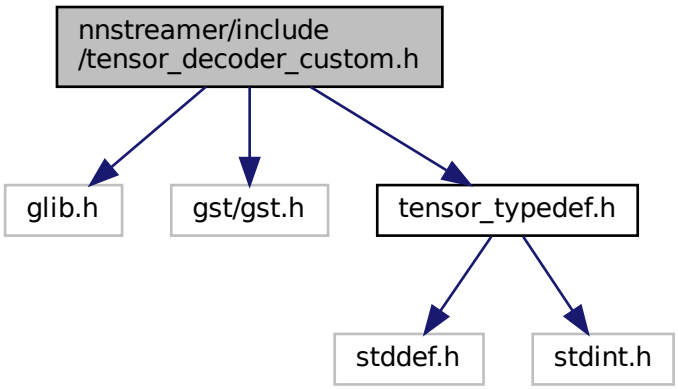
9.81 nnstreamer/include/tensor_decoder_custom.h File Reference

NNStreamer APIs for tensor_decoder custom condition.

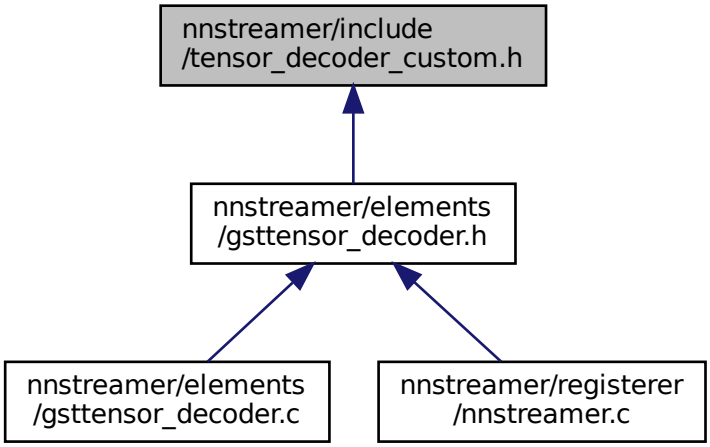
```
#include <glib.h>
#include <gst/gst.h>
```

```
#include "tensor_typedef.h"
```

Include dependency graph for tensor_decoder_custom.h:



This graph shows which files directly or indirectly include this file:



Functions

- int nnstreamer_decoder_custom_register (const gchar *name, tensor_decoder_custom func, void *data)

  Register the custom callback function.
- int nnstreamer_decoder_custom_unregister (const gchar *name)

  Unregister the custom callback function.

Variables

- G_BEGIN_DECLS typedef int(∗ tensor_decoder_custom)(const GstTensorMemory ∗input, const GstTensorsConfig ∗config, void ∗data, GstBuffer ∗out_buf)

  Decode from tensors to media as customized operation.

9.81.1 Detailed Description

NNStreamer APIs for tensor_decoder custom condition.

GStreamer/NNStreamer Tensor-Decoder Copyright (C) 2021 Gichan Jang gichan2.jang@samsung.com

Date

22 Mar 2021

See also

https://github.com/nnstreamer/nnstreamer

Author

Gichan Jang gichan2.jang@samsung.com

Bug No known bugs except for NYI items

9.81.2 Function Documentation

9.81.2.1 nnstreamer_decoder_custom_register()

```
int nnstreamer_decoder_custom_register (
            const gchar * name,
            tensor_decoder_custom func,
            void * data )
```

Register the custom callback function.

Parameters

in	*name*	The name of tensor_decoder custom callback function.
in	*func*	The custom condition function body
	[in/out]	data The internal data for the function

Returns

0 if success. -ERRNO if error.

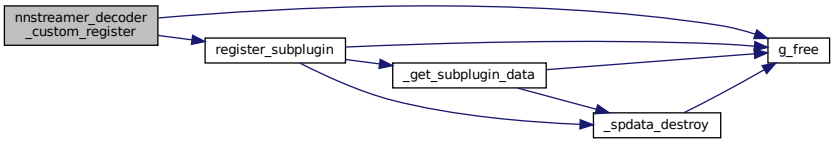
Register the custom callback function.

Returns

0 if success. -ERRNO if error.

Definition at line 972 of file gsttensor_decoder.c.

Here is the call graph for this function:



9.81.2.2 nnstreamer_decoder_custom_unregister()

```
int nnstreamer_decoder_custom_unregister (
            const gchar * name )
```

Unregister the custom callback function.

Parameters

in	*name*	The registered name of tensor_decoder custom callback function.

Returns

0 if success. -ERRNO if error.

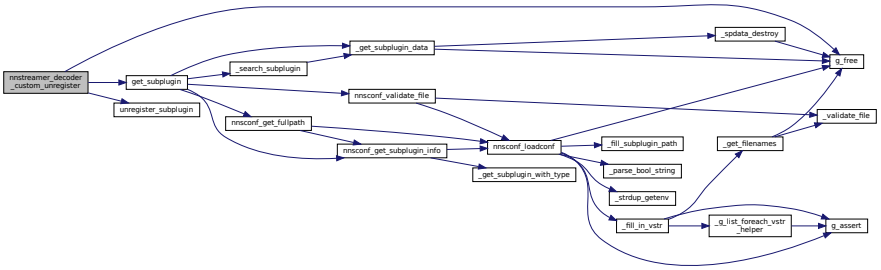
Unregister the custom callback function.

Returns

0 if success. -ERRNO if error.

Definition at line 998 of file gsttensor_decoder.c.

Here is the call graph for this function:



9.81.3 Variable Documentation

9.81.3.1 tensor_decoder_custom

```
G_BEGIN_DECLS typedef int(* tensor_decoder_custom) (const GstTensorMemory *input, const GstTensorsConfig
*config, void *data, GstBuffer *out_buf)
```

Decode from tensors to media as customized operation.

Parameters

in	*input*	the input memory containing tensors
in	*config*	input tensors config
in	*data*	private data for the callback
out	*output*	buffer filled by user

Returns

0 if success. -ERRNO if error.

Definition at line 32 of file tensor_decoder_custom.h.

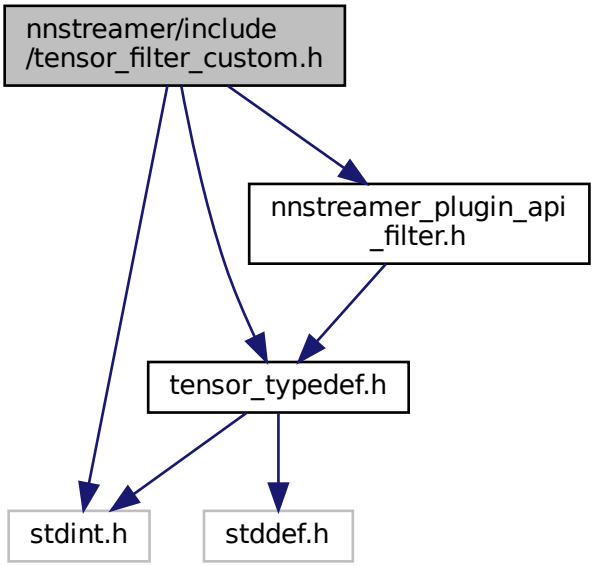
9.82 nnstreamer/include/tensor_filter_custom.h File Reference

Custom tensor post-processing interface for NNStreamer suite for post-processing code developers.

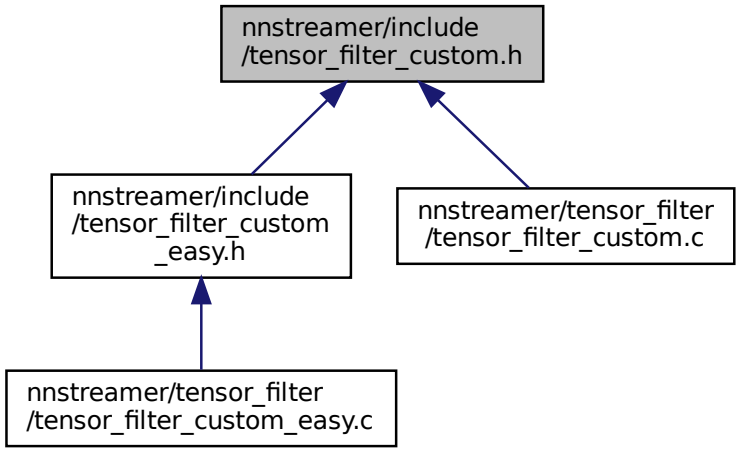
```
#include <stdint.h>
#include "tensor_typedef.h"
```

```
#include "nnstreamer_plugin_api_filter.h"
```

Include dependency graph for tensor_filter_custom.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct _NNStreamer_custom_class

  Custom Filter Class.

Typedefs

- typedef void *(* NNS_custom_init_func) (const GstTensorFilterProperties *prop)

  A function that is called before calling other functions.
- typedef void(* NNS_custom_exit_func) (void *private_data, const GstTensorFilterProperties *prop)

  A function that is called after calling other functions, when it's ready to close.
- typedef int(* NNS_custom_get_input_dimension) (void *private_data, const GstTensorFilterProperties *prop, GstTensorsInfo *info)

  Get input tensor type.
- typedef int(* NNS_custom_get_output_dimension) (void *private_data, const GstTensorFilterProperties *prop, GstTensorsInfo *info)

  Get output tensor type.
- typedef int(* NNS_custom_set_input_dimension) (void *private_data, const GstTensorFilterProperties *prop, const GstTensorsInfo *in_info, GstTensorsInfo *out_info)

  Set input dim by framework. Let custom plugin set output dim accordingly.
- typedef int(* NNS_custom_invoke) (void *private_data, const GstTensorFilterProperties *prop, const GstTensorMemory *input, GstTensorMemory *output)

  Invoke the "main function". Without allocating output buffer. (fill in the given output buffer)
- typedef int(* NNS_custom_allocate_invoke) (void *private_data, const GstTensorFilterProperties *prop, const GstTensorMemory *input, GstTensorMemory *output)

  Invoke the "main function". Without allocating output buffer. (fill in the given output buffer)
- typedef void(* NNS_custom_destroy_notify) (void *data)

  It's a post-processing method about the used data pointer if it has been allocated at custom filter.
- typedef struct _NNStreamer_custom_class NNStreamer_custom_class

Variables

- NNStreamer_custom_class * NNStreamer_custom

  A custom filter MUST define NNStreamer_custom. This object represents the custom filter itself.

9.82.1 Detailed Description

Custom tensor post-processing interface for NNStreamer suite for post-processing code developers.

GStreamer Tensor_Filter, Customized Module Copyright (C) 2018 MyungJoo Ham myungjoo.ham@samsung.↩com

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Library General Public License as published by the Free Software Foundation; version 2.1 of the License.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Library General Public License for more details.

Date

01 Jun 2018

See also

http://github.com/nnstreamer/nnstreamer

Author

MyungJoo Ham myungjoo.ham@samsung.com

Bug No known bugs except for NYI items

How To for NNdevelopers:

1. Define struct, "NNStreamer_custom", with the functions defined.
2. Compile as a shared object. (.so in Linux)
3. Use NNStreamer (tensor_filter framework=custom, model=FILEPATH_OF_YOUR_SO.so, ...)

To Packagers:

This file is to be packaged as "devel" package for NN developers.

9.82.2 Typedef Documentation

9.82.2.1 NNS_custom_allocate_invoke

```
typedef int(* NNS_custom_allocate_invoke) (void *private_data, const GstTensorFilterProperties
*prop, const GstTensorMemory *input, GstTensorMemory *output)
```

Invoke the "main function". Without allocating output buffer. (fill in the given output buffer)

Parameters

in	*private_data*	The pointer returned by NNStreamer_custom_init.
in	*prop*	GstTensorFilter's property values. Do not change its values.
in	*input*	The array of input tensors, each tensor size = dim_1 x dim_2 x .. x dim_n x typesize, allocated by caller
out	*output*	The array of output tensors, each tensor size = dim_1 x dim_2 x .. x dim_n x typesize, the memory block for output tensor should be allocated. (data in GstTensorMemory)

Note

rank limit (NNS_TENSOR_RANK_LIMIT) and typesize (tensor_element_size) defined in tensor_typedef.h

Returns

0 if success

Definition at line 111 of file tensor_filter_custom.h.

9.82.2.2 NNS_custom_destroy_notify

```
typedef void(* NNS_custom_destroy_notify) (void *data)
```

It's a post-processing method about the used data pointer if it has been allocated at custom filter.

Parameters

in	*data*	the data element.

Definition at line 118 of file tensor_filter_custom.h.

9.82.2.3 NNS_custom_exit_func

```
typedef void(* NNS_custom_exit_func) (void *private_data, const GstTensorFilterProperties
*prop)
```

A function that is called after calling other functions, when it's ready to close.

Parameters

in	*private_data*	If you have allocated *private_data at init, free it here.
in	*prop*	GstTensorFilter's property values. Do not change its values.

Definition at line 53 of file tensor_filter_custom.h.

9.82.2.4 NNS_custom_get_input_dimension

```
typedef int(* NNS_custom_get_input_dimension) (void *private_data, const GstTensorFilterProperties
*prop, GstTensorsInfo *info)
```

Get input tensor type.

Parameters

in	*private_data*	The pointer returned by NNStreamer_custom_init.
in	*prop*	GstTensorFilter's property values. Do not change its values.
out	*info*	Structure for tensor info.

Returns

0 if success

Definition at line 63 of file tensor_filter_custom.h.

9.82.2.5 NNS_custom_get_output_dimension

```
typedef int(* NNS_custom_get_output_dimension) (void *private_data, const GstTensorFilterProperties
*prop, GstTensorsInfo *info)
```

Get output tensor type.

Parameters

in	*private_data*	The pointer returned by NNStreamer_custom_init.
in	*prop*	GstTensorFilter's property values. Do not change its values.
out	*info*	Structure for tensor info.

Returns

0 if success

Definition at line 73 of file tensor_filter_custom.h.

9.82.2.6 NNS_custom_init_func

```
typedef void*(* NNS_custom_init_func) (const GstTensorFilterProperties *prop)
```

A function that is called before calling other functions.

Parameters

in	*prop*	GstTensorFilter's property values. Do not change its values.

Returns

The returned pointer will be passed to other functions as "private_data".

Definition at line 46 of file tensor_filter_custom.h.

9.82.2.7 NNS_custom_invoke

```
typedef int(* NNS_custom_invoke) (void *private_data, const GstTensorFilterProperties *prop,
const GstTensorMemory *input, GstTensorMemory *output)
```

Invoke the "main function". Without allocating output buffer. (fill in the given output buffer)

Parameters

in	*private_data*	The pointer returned by NNStreamer_custom_init.
in	*prop*	GstTensorFilter's property values. Do not change its values.
in	*input*	The array of input tensors, each tensor size = dim_1 x dim_2 x .. x dim_n x typesize, allocated by caller
out	*output*	The array of output tensors, each tensor size = dim_1 x dim_2 x .. x dim_n x typesize, allocated by caller

Note

rank limit (NNS_TENSOR_RANK_LIMIT) and typesize (tensor_element_size) defined in tensor_typedef.h

Returns

0 if success

Definition at line 99 of file tensor_filter_custom.h.

9.82.2.8 NNS_custom_set_input_dimension

```
typedef int(* NNS_custom_set_input_dimension) (void *private_data, const GstTensorFilterProperties
*prop, const GstTensorsInfo *in_info, GstTensorsInfo *out_info)
```

Set input dim by framework. Let custom plugin set output dim accordingly.

Parameters

in	*private_data*	The pointer returned by NNStreamer_custom_init.
in	*prop*	GstTensorFilter's property values. Do not change its values.
in	*in_info*	Input tensor info designated by the GStreamer framework. Note that this is not a fixed value and GStreamer may try different values during pad-cap negotiations.
out	*out_info*	Output tensor info according to the input tensor info.

@caution Do not fix internal values based on this call. GStreamer may call this function repeatedly with different values during pad-cap negotiations. Fix values when invoke is finally called.

Definition at line 87 of file tensor_filter_custom.h.

9.82.2.9 NNStreamer_custom_class

```
typedef struct _NNStreamer_custom_class NNStreamer_custom_class
```

Definition at line 136 of file tensor_filter_custom.h.

9.82.3 Variable Documentation

9.82.3.1 NNStreamer_custom

```
NNStreamer_custom_class* NNStreamer_custom
```

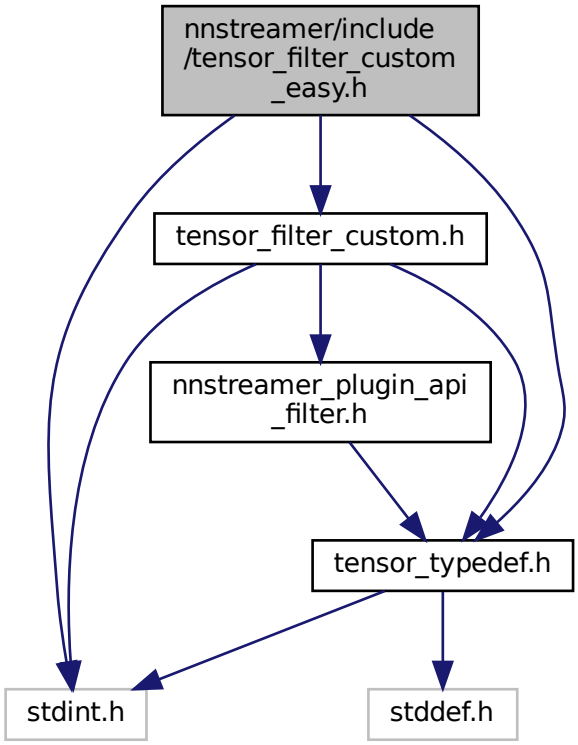
A custom filter MUST define NNStreamer_custom. This object represents the custom filter itself.

9.83 nnstreamer/include/tensor_filter_custom_easy.h File Reference

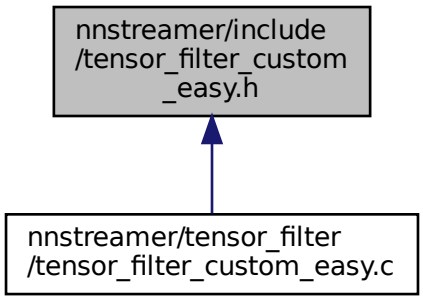
Custom tensor processing interface for simple functions.

```
#include <stdint.h>
#include "tensor_typedef.h"
#include "tensor_filter_custom.h"
```

Include dependency graph for tensor_filter_custom_easy.h:



This graph shows which files directly or indirectly include this file:



Typedefs

- typedef int(∗ NNS_custom_invoke_dynamic) (void ∗private_data, const GstTensorsInfo ∗in_info, GstTensorsInfo ∗out_info, const GstTensorMemory ∗input, GstTensorMemory ∗output)

  Invoke the "main function" with flexible input and output. Output tensor memory should be allocated.

Functions

- G_BEGIN_DECLS int NNS_custom_easy_register (const char ∗modelname, NNS_custom_invoke func, void ∗data, const GstTensorsInfo ∗in_info, const GstTensorsInfo ∗out_info)

  Register the custom-easy tensor function.
- int NNS_custom_easy_dynamic_register (const char ∗modelname, NNS_custom_invoke_dynamic func, void ∗data, const GstTensorsInfo ∗in_info)

  Register the custom-easy tensor function for dynamic invoke.
- int NNS_custom_easy_unregister (const char ∗modelname)

  Unregister the custom-easy tensor function.

9.83.1 Detailed Description

Custom tensor processing interface for simple functions.

GStreamer Tensor_Filter, Customized Module, Easy Mode Copyright (C) 2019 MyungJoo Ham myungjoo.↩ham@samsung.com

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Library General Public License as published by the Free Software Foundation; version 2.1 of the License.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Library General Public License for more details.

Date

24 Oct 2019

See also

http://github.com/nnstreamer/nnstreamer

Author

MyungJoo Ham myungjoo.ham@samsung.com

Bug No known bugs except for NYI items

How To for NNdevelopers:

Case 1. Provide the function as a shared object for other apps.

1. Define struct, "NNStreamer_custom_easy", with the functions defined.
2. Call NNS_custom_easy_register() at a global init/constructor function.
3. Compile as a shared object. (.so in Linux) And install at the custom-filter path.
4. Use NNStreamer (tensor_filter framework=custom_easy, model=${modelname}, custom=${_FILEPATH_O↩ F_YOUR_SO} ...)
5. Note that you may register multiple models (functions) with a single .so.

Case 2. Define the function in the app.

1. Define struct, "NNStreamer_custom_easy", with the functions defined.
2. Register the struct with "NNS_custom_easy_register" API.
3. Construct the nnstreamer pipeline and execute it in the app.

Note that this does not support flexible dimensions.

To Packagers:

This file is to be packaged as "devel" package for NN developers.

9.83.2 Typedef Documentation

9.83.2.1 NNS_custom_invoke_dynamic

```
typedef int(* NNS_custom_invoke_dynamic) (void *private_data, const GstTensorsInfo *in_info,
GstTensorsInfo *out_info, const GstTensorMemory *input, GstTensorMemory *output)
```

Invoke the "main function" with flexible input and output. Output tensor memory should be allocated.

Parameters

	[in/out]	private_data A subplugin may save its internal private data here. The subplugin is responsible for alloc/free of this pointer.
in	*info*	structure of input tensors info
out	*info*	structure of output tensors info. The subplugin should fill this info.
in	*input*	The array of input tensors. Allocated and filled by tensor_filter/main
out	*output*	The array of output tensors. The subplugin should allocate the memory block for output tensor. (data in GstTensorMemory)

Note

rank limit (NNS_TENSOR_RANK_LIMIT) and typesize (tensor_element_size) defined in tensor_typedef.h

Returns

0 if success

Definition at line 76 of file tensor_filter_custom_easy.h.

9.83.3 Function Documentation

9.83.3.1 NNS_custom_easy_dynamic_register()

```
int NNS_custom_easy_dynamic_register (
            const char * modelname,
            NNS_custom_invoke_dynamic func,
            void * data,
            const GstTensorsInfo * in_info )
```

Register the custom-easy tensor function for dynamic invoke.

Parameters

in	*modelname*	The name of custom-easy tensor function.
in	*func*	The tensor function body
	[in/out]	private_data The internal data for the function
in	*in_info*	Input tensor metadata.

Note

NNS_custom_invoke_dynamic defined in tensor_filter_custom.h Output buffers should be allocated in the invoke function.

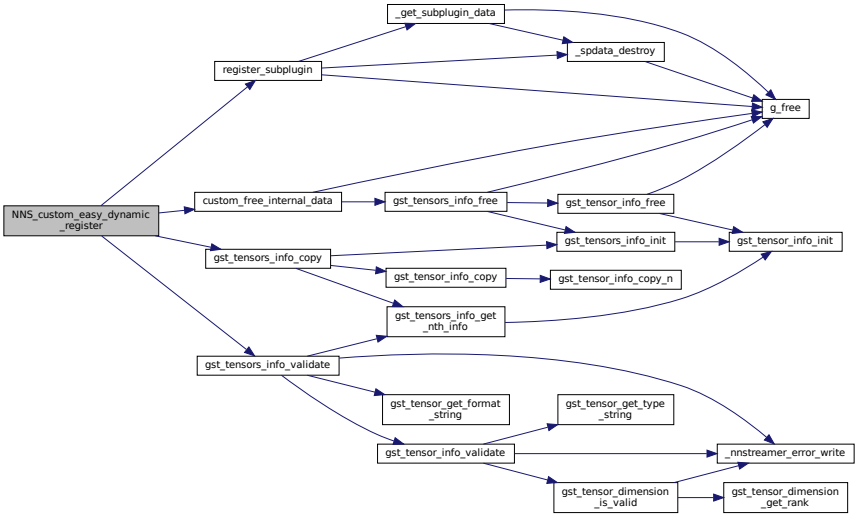
Register the custom-easy tensor function for dynamic invoke.

Returns

0 if success. -ERRNO if error.

Definition at line 112 of file tensor_filter_custom_easy.c.

Here is the call graph for this function:



9.83.3.2 NNS_custom_easy_register()

```
G_BEGIN_DECLS int NNS_custom_easy_register (
            const char * modelname,
            NNS_custom_invoke func,
            void * data,
            const GstTensorsInfo * in_info,
            const GstTensorsInfo * out_info )
```

Register the custom-easy tensor function.

Parameters

in	*modelname*	The name of custom-easy tensor function.
in	*func*	The tensor function body
	[in/out]	private_data The internal data for the function
in	*in_info*	Input tensor metadata.
in	*out_info*	Output tensor metadata

Note

NNS_custom_invoke defined in tensor_filter_custom.h Output buffers for func are preallocated.

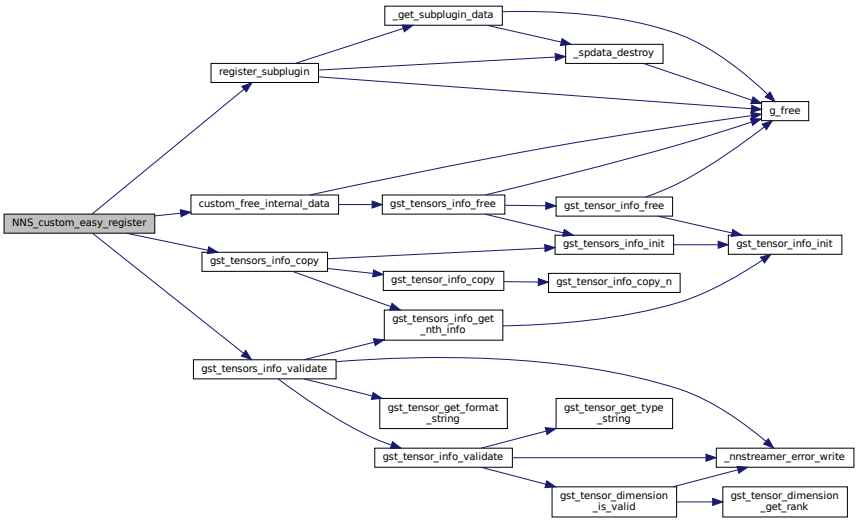
Register the custom-easy tensor function.

Returns

0 if success. -ERRNO if error.

Definition at line 76 of file tensor_filter_custom_easy.c.

Here is the call graph for this function:



9.83.3.3 NNS_custom_easy_unregister()

```
int NNS_custom_easy_unregister (
            const char * modelname )
```

Unregister the custom-easy tensor function.

Parameters

in	*modelname*	The registered name of custom-easy tensor function.

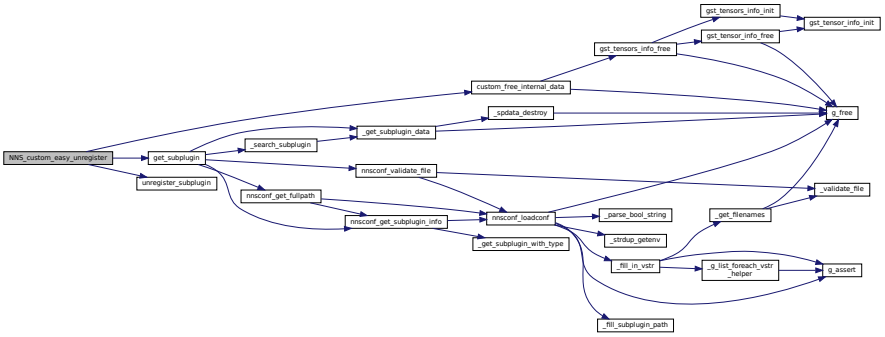
Returns

0 if success, non-zero if error

0 if success. -EINVAL if invalid model name.

Definition at line 144 of file tensor_filter_custom_easy.c.

Here is the call graph for this function:

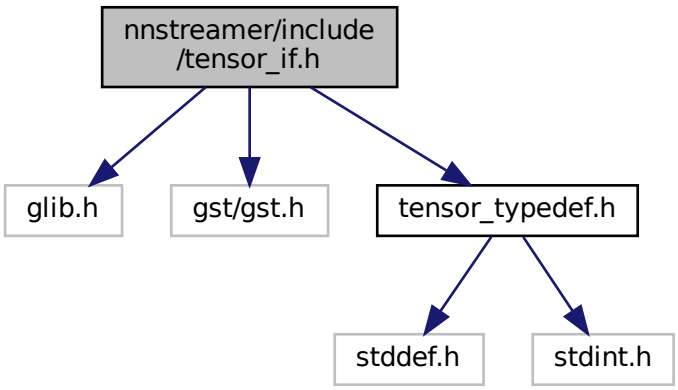


9.84 nnstreamer/include/tensor_if.h File Reference

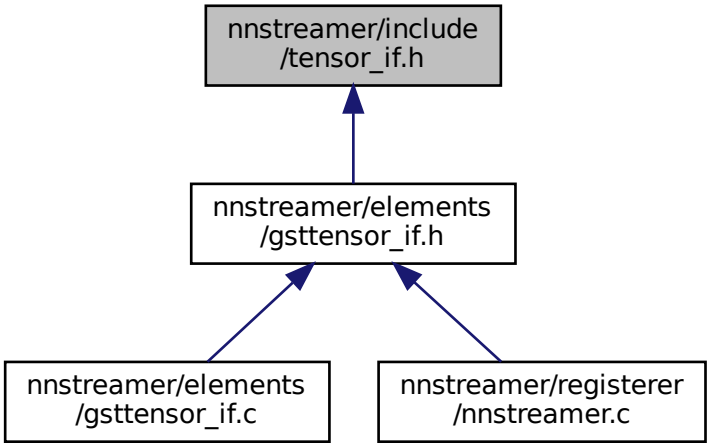
NNStreamer APIs for tensor_if custom condition.

```
#include <glib.h>
#include <gst/gst.h>
#include "tensor_typedef.h"
```

Include dependency graph for tensor_if.h:



This graph shows which files directly or indirectly include this file:



Functions

- int nnstreamer_if_custom_register (const gchar *name, tensor_if_custom func, void *data)

  Register the custom callback function.
- int nnstreamer_if_custom_unregister (const gchar *name)

  Unregister the custom callback function.

Variables

- G_BEGIN_DECLS typedef gboolean(* tensor_if_custom)(const GstTensorsInfo *info, const GstTensorMemory *input, void *user_data, gboolean *result)

  Calls the user defined conditions function.

9.84.1 Detailed Description

NNStreamer APIs for tensor_if custom condition.

GStreamer/NNStreamer Tensor-IF Copyright (C) 2020 Gichan Jang gichan2.jang@samsung.com

Date

28 Oct 2020

See also

https://github.com/nnstreamer/nnstreamer

Author

Gichan Jang gichan2.jang@samsung.com

Bug No known bugs except for NYI items

9.84.2 Function Documentation

9.84.2.1 nnstreamer_if_custom_register()

```
int nnstreamer_if_custom_register (
            const gchar * name,
            tensor_if_custom func,
            void * data )
```

Register the custom callback function.

Parameters

in	*name*	The name of tensor_if custom callback function.
in	*func*	The custom condition function body
	[in/out]	data The internal data for the function

Returns

0 if success. -ERRNO if error.

Note

GstElement`tensor_if_h` can be get with `gst_bin_get_by_name` from the pipeline. e.g.,) pipeline description: ... ! tensor_if name=tif (... properties ...) ! ... GstElement *tensor_if_h = gst_bin_get_by_name (GST_BIN (pipeline), "tif");

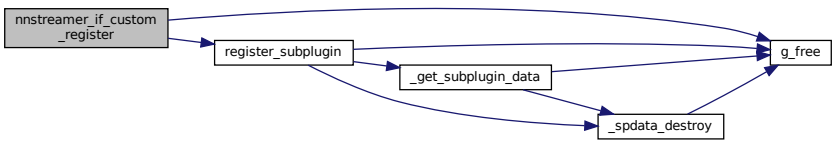
Register the custom callback function.

Returns

0 if success. -ERRNO if error.

Definition at line 1023 of file gsttensor_if.c.

Here is the call graph for this function:



9.84.2.2 nnstreamer_if_custom_unregister()

```
int nnstreamer_if_custom_unregister (
            const gchar * name )
```

Unregister the custom callback function.

Parameters

in	*name*	The registered name of tensor_if custom callback function.

Returns

0 if success. -ERRNO if error.

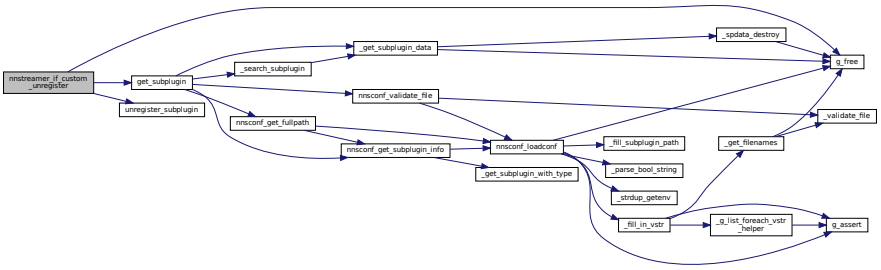
Unregister the custom callback function.

Returns

0 if success. -ERRNO if error.

Definition at line 1049 of file gsttensor_if.c.

Here is the call graph for this function:



9.84.3 Variable Documentation

9.84.3.1 tensor_if_custom

```
G_BEGIN_DECLS typedef gboolean(* tensor_if_custom) (const GstTensorsInfo *info, const GstTensorMemory
*input, void *user_data, gboolean *result)
```

Calls the user defined conditions function.

SECTION:element-tensor_if

How To for NNdevelopers:

Define the function in the app.

1. Define struct, "tensor_if_custom", with the functions defined.
2. Register the struct with "nnstreamer_if_custom_register" API.
3. Construct the nnstreamer pipeline and execute it in the app.

Usage example of the tensor_if custom condition

```
// Define custom callback function describing the condition
gboolean tensor_if_custom_cb (const GstTensorsInfo *info,
    const GstTensorMemory * input, gboolean * result) {
  // Describe the conditions and pass the results
}
...
// Register custom callback function describing the condition
nnstreamer_if_custom_register ("tifx", tensor_if_custom_cb, NULL);
...
// Use the condition in a pipeline.
// E.g., Pipeline of " ... ! tensor_if compared-value=CUSTOM compared-value-option=tifx then=TENSORPICK
       then-option=0 else=TENSORPICK else-option=1 ! ... "
...
// After everything is done.
nnstreamer_if_custom_unregister ("tifx");
```

Parameters

in	*info*	input tensors info
in	*input*	memory containing input tensor data
in	*user_data*	private data for the callback
out	*result*	result of the user defined condition

Returns

TRUE if there is no error.

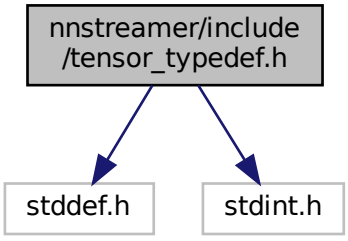
Definition at line 63 of file tensor_if.h.

9.85 nnstreamer/include/tensor_typedef.h File Reference

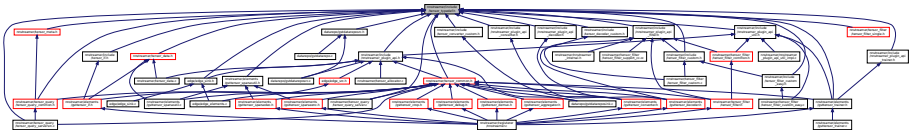
Common header file for NNStreamer, the GStreamer plugin for neural networks.

```
#include <stddef.h>
#include <stdint.h>
```

Include dependency graph for tensor_typedef.h:



This graph shows which files directly or indirectly include this file:



Classes

- union tensor_element

  To make the code simple with all the types. "C++ Template"-like.
- struct GstTensorMemory

  The unit of each data tensors. It will be used as an input/output tensor of other/tensors.
- struct GstTensorInfo

  Internal data structure for tensor info.
- struct GstTensorsInfo

  Internal meta data exchange format for a other/tensors instance.
- struct GstTensorsConfig

  Internal data structure for configured tensors info (for other/tensors).
- struct GstSparseTensorInfo

  Internal data structure for sparse tensor info.
- struct GstTensorMetaInfo

  Data structure to describe a tensor data. This represents the basic information of a memory block for tensor stream.

Macros

- #define NNS_TENSOR_RANK_LIMIT (16)
- #define NNS_TENSOR_SIZE_LIMIT (256)

  The number of tensors NNStreamer supports is 256. The max memories of gst-buffer is 16 (See NNS_TENSOR↩ _MEMORY_MAX). Internally NNStreamer handles the memories as tensors. If the number of tensors is larger than 16, we modify the last memory and combine tensors into the memory.
- #define NNS_TENSOR_SIZE_LIMIT_STR "256"
- #define NNS_TENSOR_MEMORY_MAX (16)

  This value, 16, can be checked with gst_buffer_get_max_memory(), which is GST_BUFFER_MEM_MAX in gstreamer/gstbuffer.c. We redefined the value because GST_BUFFER_MEM_MAX is not exported and we need static value. To modify (increase) this value, you need to update gstreamer/gstbuffer.c as well.
- #define NNS_TENSOR_SIZE_EXTRA_LIMIT (NNS_TENSOR_SIZE_LIMIT - NNS_TENSOR_MEMORY_MAX)

  Max number of extra tensors.
- #define NNS_MIMETYPE_TENSOR "other/tensor"
- #define NNS_MIMETYPE_TENSORS "other/tensors"
- #define GST_TENSOR_NUM_TENSORS_RANGE "(int) [1, " NNS_TENSOR_SIZE_LIMIT_STR "]"
- #define GST_TENSOR_RATE_RANGE "(fraction) [0, max]"
- #define GST_TENSOR_TYPE_ALL "{ float16, float32, float64, int64, uint64, int32, uint32, int16, uint16, int8, uint8 }"

  Possible tensor element types.
- #define GST_TENSOR_FORMAT_ALL "{ static, flexible, sparse }"

  Possible tensor formats.
- #define GST_TENSOR_CAP_DEFAULT

  Default static capability for other/tensor.
- #define GST_TENSORS_CAP_MAKE(fmt)

  Caps string for the caps template of tensor stream. format should be a string that describes the data format, or possible formats of incoming tensor.
- #define GST_TENSORS_CAP_WITH_NUM(num)

  Caps string for the caps template (other/tensors, static tensor stream with fixed number of tensors). num should be a string format that describes the number of tensors, or the range of incoming tensors. The types and dimensions of tensors should be described for caps negotiation.
- #define GST_TENSORS_CAP_DEFAULT GST_TENSORS_CAP_WITH_NUM (GST_TENSOR_NUM_TENSORS_RANGE)

  Caps string for the caps template of static tensor stream.
- #define GST_TENSORS_FLEX_CAP_DEFAULT GST_TENSORS_CAP_MAKE ("flexible")

Caps string for the caps template of flexible tensors. This mimetype handles non-static, flexible tensor stream without specifying the data type and shape of the tensor. The maximum number of tensors in a buffer is 16 (NNS_TENSO↩ R_SIZE_LIMIT).

- #define GST_TENSORS_SPARSE_CAP_DEFAULT GST_TENSORS_CAP_MAKE ("sparse")

  Caps string for the caps template of sparse tensors. This mimetype handles non-static, sparse tensor stream without specifying the data type and shape of the tensor. The maximum number of tensors in a buffer is 16 (NNS_TENSO↩ R_SIZE_LIMIT).

Typedefs

- typedef enum _nns_tensor_type tensor_type

  Possible data element types of other/tensor.
- typedef enum _nns_media_type media_type

  Float16 compiler extension support.
- typedef enum _tensor_format tensor_format

  Data format of tensor stream in the pipeline.
- typedef enum _nns_tensor_layout tensor_layout

  Tensor layout format for other/tensor.
- typedef uint32_t tensor_dim[NNS_TENSOR_RANK_LIMIT]

Enumerations

- enum _nns_tensor_type {
  _NNS_INT32 = 0, _NNS_UINT32, _NNS_INT16, _NNS_UINT16,
  _NNS_INT8, _NNS_UINT8, _NNS_FLOAT64, _NNS_FLOAT32,
  _NNS_INT64, _NNS_UINT64, _NNS_FLOAT16, _NNS_END }

  Possible data element types of other/tensor.
- enum _nns_media_type {
  _NNS_MEDIA_INVALID = -1, _NNS_VIDEO = 0, _NNS_AUDIO = 1, _NNS_TEXT = 2,
  _NNS_OCTET = 3, _NNS_TENSOR = 4, _NNS_MEDIA_ANY = 0x1000 }

  Float16 compiler extension support.
- enum _tensor_format { _NNS_TENSOR_FORMAT_STATIC = 0, _NNS_TENSOR_FORMAT_FLEXIBLE,
  _NNS_TENSOR_FORMAT_SPARSE, _NNS_TENSOR_FORMAT_END }

  Data format of tensor stream in the pipeline.
- enum _nns_tensor_layout { _NNS_LAYOUT_ANY = 0, _NNS_LAYOUT_NHWC, _NNS_LAYOUT_NCHW,
  _NNS_LAYOUT_NONE }

  Tensor layout format for other/tensor.

9.85.1 Detailed Description

Common header file for NNStreamer, the GStreamer plugin for neural networks.

NNStreamer Common Header, Typedef part, for export as devel package. Copyright (C) 2018 MyungJoo Ham myungjoo.ham@samsung.com

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Library General Public License as published by the Free Software Foundation; version 2.1 of the License.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Library General Public License for more details.

Date

01 Jun 2018

See also

http://github.com/nnstreamer/nnstreamer

Author

MyungJoo Ham myungjoo.ham@samsung.com

Bug No known bugs except for NYI items

To Packagers:

This file is to be packaged as "devel" package for NN developers.

9.85.2 Macro Definition Documentation

9.85.2.1 GST_TENSOR_CAP_DEFAULT

```
#define GST_TENSOR_CAP_DEFAULT
```

Value:

```
    NNS_MIMETYPE_TENSOR ", " \
    "framerate = " GST_TENSOR_RATE_RANGE
```

Default static capability for other/tensor.

Definition at line 78 of file tensor_typedef.h.

9.85.2.2 GST_TENSOR_FORMAT_ALL

```
#define GST_TENSOR_FORMAT_ALL "{ static, flexible, sparse }"
```

Possible tensor formats.

Definition at line 73 of file tensor_typedef.h.

9.85.2.3 GST_TENSOR_NUM_TENSORS_RANGE

```
#define GST_TENSOR_NUM_TENSORS_RANGE "(int) [ 1, " NNS_TENSOR_SIZE_LIMIT_STR " ]"
```

Definition at line 62 of file tensor_typedef.h.

9.85.2.4 GST_TENSOR_RATE_RANGE

```
#define GST_TENSOR_RATE_RANGE "(fraction) [ 0, max ]"
```

Definition at line 63 of file tensor_typedef.h.

9.85.2.5 GST_TENSOR_TYPE_ALL

```
#define GST_TENSOR_TYPE_ALL "{ float16, float32, float64, int64, uint64, int32, uint32, int16,
uint16, int8, uint8 }"
```

Possible tensor element types.

Definition at line 68 of file tensor_typedef.h.

9.85.2.6 GST_TENSORS_CAP_DEFAULT

```
#define GST_TENSORS_CAP_DEFAULT GST_TENSORS_CAP_WITH_NUM (GST_TENSOR_NUM_TENSORS_RANGE)
```

Caps string for the caps template of static tensor stream.

Definition at line 115 of file tensor_typedef.h.

9.85.2.7 GST_TENSORS_CAP_MAKE

```
#define GST_TENSORS_CAP_MAKE(
               fmt )
```

Value:

```
NNS_MIMETYPE_TENSORS ", " \
"format = (string) " fmt ", " \
"framerate = " GST_TENSOR_RATE_RANGE
```

Caps string for the caps template of tensor stream. format should be a string that describes the data format, or possible formats of incoming tensor.

type should be one of types in GST_TENSOR_TYPE_ALL "type = (string) uint8" dimension should be a formatted string with rank NNS_TENSOR_RANK_LIMIT "dimension = (string) dim1:dim2:dim3:dim4" If the data format is static, another tensor information should be described in caps.

- num_tensors: The number of tensors in a GstBuffer.
- types: The list of data type in the tensor. type should be one of types in GST_TENSOR_TYPE_ALL. (types=(string)"type1,type2,type3")
- dimensions: The list of dimension in the tensor. dimension should be a formatted string with rank NNS_TE↩ NSOR_RANK_LIMIT. (dimensions=(string)"dim1:dim2:dim3:dim4,dim1:dim2:dim3:dim4")

Definition at line 97 of file tensor_typedef.h.

9.85.2.8 GST_TENSORS_CAP_WITH_NUM

```
#define GST_TENSORS_CAP_WITH_NUM(
              num )
```

Value:

```
NNS_MIMETYPE_TENSORS ", " \
"format = (string) static, num_tensors = " num ", " \
"framerate = " GST_TENSOR_RATE_RANGE
```

Caps string for the caps template (other/tensors, static tensor stream with fixed number of tensors). num should be a string format that describes the number of tensors, or the range of incoming tensors. The types and dimensions of tensors should be described for caps negotiation.

Definition at line 107 of file tensor_typedef.h.

9.85.2.9 GST_TENSORS_FLEX_CAP_DEFAULT

```
#define GST_TENSORS_FLEX_CAP_DEFAULT GST_TENSORS_CAP_MAKE ("flexible")
```

Caps string for the caps template of flexible tensors. This mimetype handles non-static, flexible tensor stream without specifying the data type and shape of the tensor. The maximum number of tensors in a buffer is 16 (NNS↩ _TENSOR_SIZE_LIMIT).

Definition at line 123 of file tensor_typedef.h.

9.85.2.10 GST_TENSORS_SPARSE_CAP_DEFAULT

```
#define GST_TENSORS_SPARSE_CAP_DEFAULT GST_TENSORS_CAP_MAKE ("sparse")
```

Caps string for the caps template of sparse tensors. This mimetype handles non-static, sparse tensor stream without specifying the data type and shape of the tensor. The maximum number of tensors in a buffer is 16 (NNS↩ _TENSOR_SIZE_LIMIT).

Definition at line 131 of file tensor_typedef.h.

9.85.2.11 NNS_MIMETYPE_TENSOR

```
#define NNS_MIMETYPE_TENSOR "other/tensor"
```

Definition at line 59 of file tensor_typedef.h.

9.85.2.12 NNS_MIMETYPE_TENSORS

```
#define NNS_MIMETYPE_TENSORS "other/tensors"
```

Definition at line 60 of file tensor_typedef.h.

9.85.2.13 NNS_TENSOR_MEMORY_MAX

```
#define NNS_TENSOR_MEMORY_MAX (16)
```

This value, 16, can be checked with gst_buffer_get_max_memory(), which is GST_BUFFER_MEM_MAX in gstreamer/gstbuffer.c. We redefined the value because GST_BUFFER_MEM_MAX is not exported and we need static value. To modify (increase) this value, you need to update gstreamer/gstbuffer.c as well.

Definition at line 52 of file tensor_typedef.h.

9.85.2.14 NNS_TENSOR_RANK_LIMIT

```
#define NNS_TENSOR_RANK_LIMIT (16)
```

Definition at line 34 of file tensor_typedef.h.

9.85.2.15 NNS_TENSOR_SIZE_EXTRA_LIMIT

```
#define NNS_TENSOR_SIZE_EXTRA_LIMIT (NNS_TENSOR_SIZE_LIMIT - NNS_TENSOR_MEMORY_MAX)
```

Max number of extra tensors.

Definition at line 57 of file tensor_typedef.h.

9.85.2.16 NNS_TENSOR_SIZE_LIMIT

```
#define NNS_TENSOR_SIZE_LIMIT (256)
```

The number of tensors NNStreamer supports is 256. The max memories of gst-buffer is 16 (See NNS_TENSOR↩ _MEMORY_MAX). Internally NNStreamer handles the memories as tensors. If the number of tensors is larger than 16, we modify the last memory and combine tensors into the memory.

Definition at line 42 of file tensor_typedef.h.

9.85.2.17 NNS_TENSOR_SIZE_LIMIT_STR

```
#define NNS_TENSOR_SIZE_LIMIT_STR "256"
```

Definition at line 43 of file tensor_typedef.h.

9.85.3 Typedef Documentation

9.85.3.1 media_type

```
typedef enum _nns_media_type media_type
```

Float16 compiler extension support.

Possible input stream types for other/tensor.

This is related with media input stream to other/tensor. There is no restrictions for the outputs.

In order to prevent enum-mix issues between device profiles, we explicitly define numbers for each enum type.

9.85.3.2 tensor_dim

```
typedef uint32_t tensor_dim[NNS_TENSOR_RANK_LIMIT]
```

Definition at line 247 of file tensor_typedef.h.

9.85.3.3 tensor_format

```
typedef enum _tensor_format tensor_format
```

Data format of tensor stream in the pipeline.

9.85.3.4 tensor_layout

```
typedef enum _nns_tensor_layout tensor_layout
```

Tensor layout format for other/tensor.

The layout is needed by some of the element to appropriately process the data based on the axis of the channel in the data. Layout information will be currently utilized by only some of the elements (SNAP, NNFW in tensor_filter, PADDING mode in tensor_transform)

Tensor layout is not part of the capabilities of the element, and does not take part in the caps negotiation.

NONE layout implies that the layout of the data is neither NHWC nor NCHW. ' However, ANY layout implies that the layout of the provided data is not relevant.

Note

Providing tensor layout can also decide acceleration to be supported as not all the accelerators might support all the layouts (NYI).

9.85.3.5 tensor_type

```
typedef enum _nns_tensor_type tensor_type
```

Possible data element types of other/tensor.

Note

When changing tensor type, you should update related type in ML-API and protobuf/flatbuf schema also.

9.85.4 Enumeration Type Documentation

9.85.4.1 _nns_media_type

```
enum _nns_media_type
```

Float16 compiler extension support.

Possible input stream types for other/tensor.

This is related with media input stream to other/tensor. There is no restrictions for the outputs.

In order to prevent enum-mix issues between device profiles, we explicitly define numbers for each enum type.

Enumerator

_NNS_MEDIA_INVALID	Uninitialized
_NNS_VIDEO	supposedly video/x-raw
_NNS_AUDIO	supposedly audio/x-raw
_NNS_TEXT	supposedly text/x-raw
_NNS_OCTET	supposedly application/octet-stream
_NNS_TENSOR	supposedly other/tensor(s) or flexible tensor
_NNS_MEDIA_ANY	any media type (find proper external converter in tensor-converter element)

Definition at line 179 of file tensor_typedef.h.

9.85.4.2 _nns_tensor_layout

```
enum _nns_tensor_layout
```

Tensor layout format for other/tensor.

The layout is needed by some of the element to appropriately process the data based on the axis of the channel in the data. Layout information will be currently utilized by only some of the elements (SNAP, NNFW in tensor_filter, PADDING mode in tensor_transform)

Tensor layout is not part of the capabilities of the element, and does not take part in the caps negotiation.

NONE layout implies that the layout of the data is neither NHWC nor NCHW. ' However, ANY layout implies that the layout of the provided data is not relevant.

Note

Providing tensor layout can also decide acceleration to be supported as not all the accelerators might support all the layouts (NYI).

Enumerator

_NNS_LAYOUT_ANY	does not care about the data layout
_NNS_LAYOUT_NHWC	NHWC: channel last layout
_NNS_LAYOUT_NCHW	NCHW: channel first layout
_NNS_LAYOUT_NONE	NONE: none of the above defined layouts

Definition at line 220 of file tensor_typedef.h.

9.85.4.3 _nns_tensor_type

`enum _nns_tensor_type`

Possible data element types of other/tensor.

Note

When changing tensor type, you should update related type in ML-API and protobuf/flatbuf schema also.

Enumerator

_NNS_INT32	
_NNS_UINT32	
_NNS_INT16	
_NNS_UINT16	
_NNS_INT8	
_NNS_UINT8	
_NNS_FLOAT64	
_NNS_FLOAT32	
_NNS_INT64	
_NNS_UINT64	
_NNS_FLOAT16	added with nnstreamer 2.1.1-devel. If you add any operators (e.g., tensor_transform) to float16, it will either be not supported or be too inefficient.
_NNS_END	

Definition at line 138 of file tensor_typedef.h.

9.85.4.4 _tensor_format

`enum _tensor_format`

Data format of tensor stream in the pipeline.

Enumerator

_NNS_TENSOR_FORMAT_STATIC	
_NNS_TENSOR_FORMAT_FLEXIBLE	
_NNS_TENSOR_FORMAT_SPARSE	
_NNS_TENSOR_FORMAT_END	

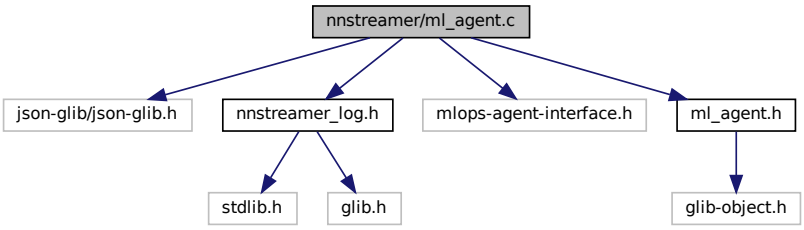
Definition at line 193 of file tensor_typedef.h.

9.86 nnstreamer/ml_agent.c File Reference

Internal helpers to make a bridge between NNS filters and the ML Agent service.

```
#include <json-glib/json-glib.h>
#include <nnstreamer_log.h>
#include <mlops-agent-interface.h>
#include "ml_agent.h"
```

Include dependency graph for ml_agent.c:



Functions

- gchar * mlagent_get_model_path_from (const GValue *val)

  Get a path of the model file from a given GValue.

Variables

- const gchar URI_SCHEME [ ] = "mlagent"
- const gchar URI_KEYWORD_MODEL [ ] = "model"
- const gchar JSON_KEY_MODEL_PATH [ ] = "path"

9.86.1 Detailed Description

Internal helpers to make a bridge between NNS filters and the ML Agent service.

Date

23 Jun 2023

See also

http://github.com/nnstreamer/nnstreamer

Author

wook16.song wook16.song@samsung.com

Bug No known bugs except for NYI items

9.86.2 Function Documentation

9.86.2.1 mlagent_get_model_path_from()

```
gchar* mlagent_get_model_path_from (
            const GValue * val )
```

Get a path of the model file from a given GValue.

Parameters

in	*val*	A pointer to a GValue holding a G_TYPE_STRING value

Returns

A newly allocated c-string representing the model file path, if the given GValue contains a valid URI Otherwise, it simply returns a duplicated (strdup'ed) c-string that the val contains.

Note

The caller should free the return c-string after using it.

Common checker for the given URI

Note

Only for the following URI formats to get the file path of the matching models are currently supported. mlagent://model/name or mlagent://model/name/version

It is required to be revised to support more scenarios that exploit the ML Agent.

Convert the given URI for a model to the file path

Todo The specification of the data layout filled in the third argument (i.e., stringified_json) by the callee is not fully decided.

Todo Parse stringified_json to get the model's path

Definition at line 33 of file ml_agent.c.

Here is the caller graph for this function:

9.86.3 Variable Documentation

9.86.3.1 JSON_KEY_MODEL_PATH

```
const gchar JSON_KEY_MODEL_PATH[] = "path"
```

Definition at line 23 of file ml_agent.c.

9.86.3.2 URI_KEYWORD_MODEL

```
const gchar URI_KEYWORD_MODEL[] = "model"
```

Definition at line 22 of file ml_agent.c.

9.86.3.3 URI_SCHEME

```
const gchar URI_SCHEME[] = "mlagent"
```

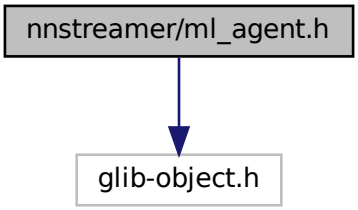
Definition at line 21 of file ml_agent.c.

9.87 nnstreamer/ml_agent.h File Reference

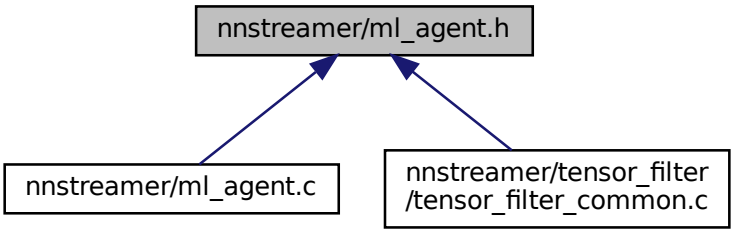
Internal header to make a bridge between NNS filters and the ML Agent service.

```
#include <glib-object.h>
```

Include dependency graph for ml_agent.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define mlagent_get_model_path_from(v) g_value_dup_string (v)

9.87.1 Detailed Description

Internal header to make a bridge between NNS filters and the ML Agent service.

Date

23 Jun 2023

See also

http://github.com/nnstreamer/nnstreamer

Author

wook16.song wook16.song@samsung.com

Bug No known bugs except for NYI items

9.87.2 Macro Definition Documentation

9.87.2.1 mlagent_get_model_path_from

```
#define mlagent_get_model_path_from(
              v ) g_value_dup_string (v)
```

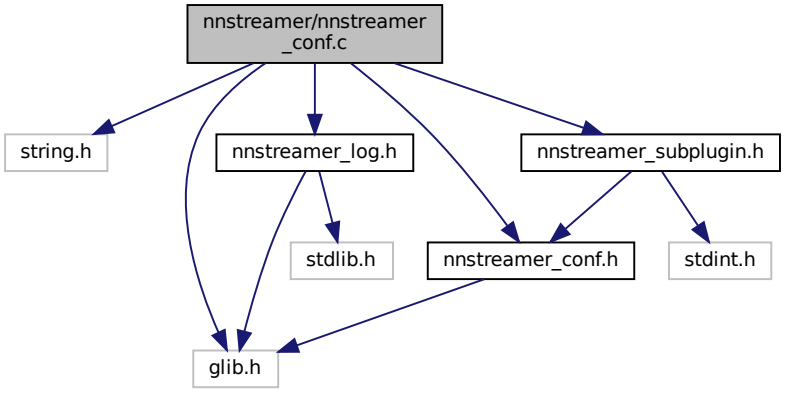
Definition at line 33 of file ml_agent.h.

9.88 nnstreamer/nnstreamer_conf.c File Reference

NNStreamer Configuration (conf file, env-var) Management.

```
#include <string.h>
#include <glib.h>
#include "nnstreamer_log.h"
#include "nnstreamer_conf.h"
#include "nnstreamer_subplugin.h"
```

Include dependency graph for nnstreamer_conf.c:



Classes

- struct subplugin_conf
- struct confdata
- struct vstr_helper

  Data structure for _g_list_foreach_vstr_helper.
- struct dump_buf

Macros

- #define NNSTREAMER_PREFIX_DECODER "libnnstreamer_decoder_"
- #define NNSTREAMER_PREFIX_FILTER "libnnstreamer_filter_"
- #define NNSTREAMER_PREFIX_CUSTOMFILTERS ""
- #define NNSTREAMER_PREFIX_CONVERTER "libnnstreamer_converter_"
- #define NNSTREAMER_PREFIX_TRAINER "libnnstreamer_trainer_"
- #define STR_BOOL(x) ((x) ? "TRUE" : "FALSE")

Enumerations

- enum conf_sources {
  CONF_SOURCE_ENVVAR = 0, CONF_SOURCE_INI = 1, CONF_SOURCE_HARDCODE = 2,
  CONF_SOURCE_EXTRA_CONF = 3,
  CONF_SOURCE_END }

Functions

- static gboolean _parse_bool_string (const gchar *strval, gboolean def)

  Parse string to get boolean value.
- static gchar * _strdup_getenv (const gchar *name)

  Private function to get strdup-ed env-var if it's valid. Otherwise, NULL.
- static gboolean _validate_file (nnsconf_type_path type, const gchar *fullpath)

  Private function to validate .so file can be added to the list.
- static gboolean _get_filenames (nnsconf_type_path type, const gchar *dir, GSList **listF, GSList **listN, guint *counter)

  Private function to fill in ".so/.dylib list" with fullpath-filenames in a directory.
- static gboolean _get_subplugin_with_type (nnsconf_type_path type, gchar ***name, gchar ***filepath)

  Private function to get sub-plugins list with type.
- static void _g_list_foreach_vstr_helper (gpointer data, gpointer user_data)

  Private function to help convert linked-list to vstr with foreach @data The element data of linked-list @user_data The struct to fill in vstr.
- static void _fill_in_vstr (gchar ***fullpath_vstr, gchar ***name_vstr, gchar *searchpath[CONF_SOURCE_END], nnsconf_type_path type)

  Private function to fill in vstr.
- static void _fill_subplugin_path (confdata *cdata, GKeyFile *key_file, conf_sources src)

  Private function to fill subplugin path.
- gboolean nnsconf_loadconf (gboolean force_reload)

  Public function defined in the header.
- const gchar * nnsconf_get_fullpath (const gchar *subpluginname, nnsconf_type_path type)

  Public function defined in the header.
- gboolean nnsconf_validate_file (nnsconf_type_path type, const gchar *fullpath)

Public function to validate sub-plugin library is available.

- const gchar * nnsconf_get_subplugin_name_prefix (nnsconf_type_path type)

  Get sub-plugin's name prefix.
- guint nnsconf_get_subplugin_info (nnsconf_type_path type, subplugin_info_s *info)

  Public function to get the list of sub-plugins name and path.
- gchar * nnsconf_get_custom_value_string (const gchar *group, const gchar *key)

  Public function defined in the header.
- gboolean nnsconf_get_custom_value_bool (const gchar *group, const gchar *key, gboolean def)

  Public function defined in the header.
- void nnsconf_dump (gchar *str, gulong size)

  Print out configurations.
- static void _foreach_custom_property (GQuark key_id, gpointer data, gpointer user_data)

  foreach callback for custom property
- void nnsconf_subplugin_dump (gchar *str, gulong size)

  Print out the information of registered sub-plugins.

Variables

- static const gchar * NNSTREAMER_ENVVAR [NNSCONF_PATH_END]
- static const gchar * NNSTREAMER_PATH [NNSCONF_PATH_END]
- static const gchar * subplugin_prefixes [ ]
- static confdata conf = { 0 }
- static GHashTable * custom_table = NULL

  Internal cache for the custom key-values.

9.88.1 Detailed Description

NNStreamer Configuration (conf file, env-var) Management.

NNStreamer Configurations / Environmental Variable Manager. Copyright (C) 2018 MyungJoo Ham myungjoo.ham@samsung.com

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Library General Public License as published by the Free Software Foundation; version 2.1 of the License.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Library General Public License for more details.

Date

26 Nov 2018

See also

https://github.com/nnstreamer/nnstreamer

Author

MyungJoo Ham myungjoo.ham@samsung.com

Bug No known bugs except for NYI items

9.88.2 Macro Definition Documentation

9.88.2.1 NNSTREAMER_PREFIX_CONVERTER

```
#define NNSTREAMER_PREFIX_CONVERTER "libnnstreamer_converter_"
```

Definition at line 42 of file nnstreamer_conf.c.

9.88.2.2 NNSTREAMER_PREFIX_CUSTOMFILTERS

```
#define NNSTREAMER_PREFIX_CUSTOMFILTERS ""
```

Definition at line 41 of file nnstreamer_conf.c.

9.88.2.3 NNSTREAMER_PREFIX_DECODER

```
#define NNSTREAMER_PREFIX_DECODER "libnnstreamer_decoder_"
```

Note that users still can place their custom filters anywhere if they designate them with the full path.

Definition at line 39 of file nnstreamer_conf.c.

9.88.2.4 NNSTREAMER_PREFIX_FILTER

```
#define NNSTREAMER_PREFIX_FILTER "libnnstreamer_filter_"
```

Definition at line 40 of file nnstreamer_conf.c.

9.88.2.5 NNSTREAMER_PREFIX_TRAINER

```
#define NNSTREAMER_PREFIX_TRAINER "libnnstreamer_trainer_"
```

Definition at line 43 of file nnstreamer_conf.c.

9.88.2.6 STR_BOOL

```
#define STR_BOOL(
              x ) ((x) ?  "TRUE" :  "FALSE")
```

Definition at line 622 of file nnstreamer_conf.c.

9.88.3 Enumeration Type Documentation

9.88.3.1 conf_sources

```
enum conf_sources
```

Enumerator

CONF_SOURCE_ENVVAR	
CONF_SOURCE_INI	
CONF_SOURCE_HARDCODE	
CONF_SOURCE_EXTRA_CONF	path from extra config file
CONF_SOURCE_END	

Definition at line 73 of file nnstreamer_conf.c.

9.88.4 Function Documentation

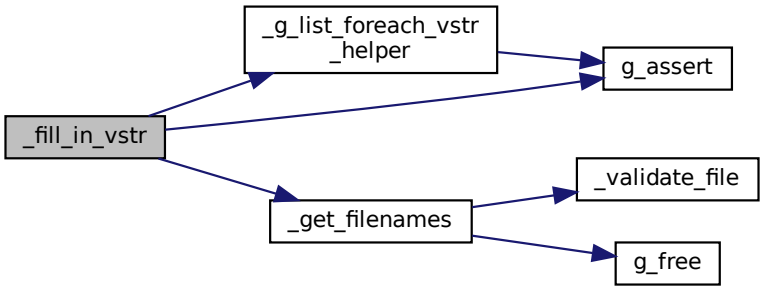
9.88.4.1 _fill_in_vstr()

```
static void _fill_in_vstr (
             gchar *** fullpath_vstr,
             gchar *** name_vstr,
             gchar * searchpath[CONF_SOURCE_END],
             nnsconf_type_path type )  [static]
```

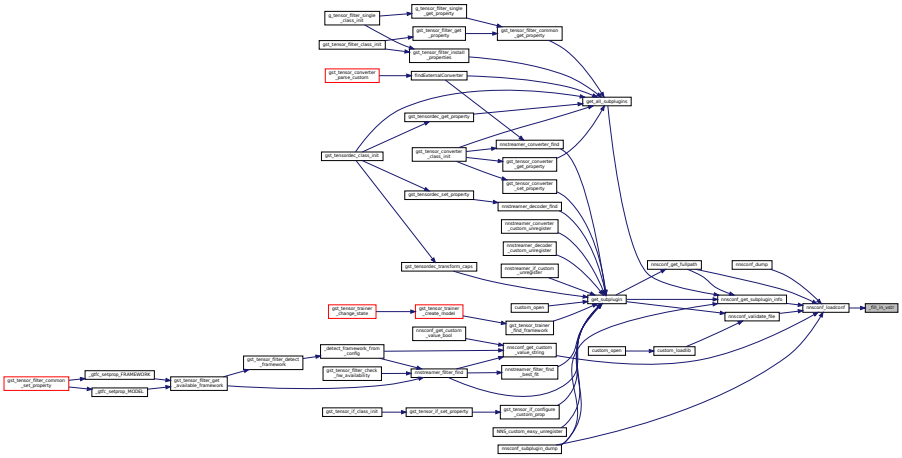
Private function to fill in vstr.

Definition at line 280 of file nnstreamer_conf.c.

Here is the call graph for this function:



Here is the caller graph for this function:



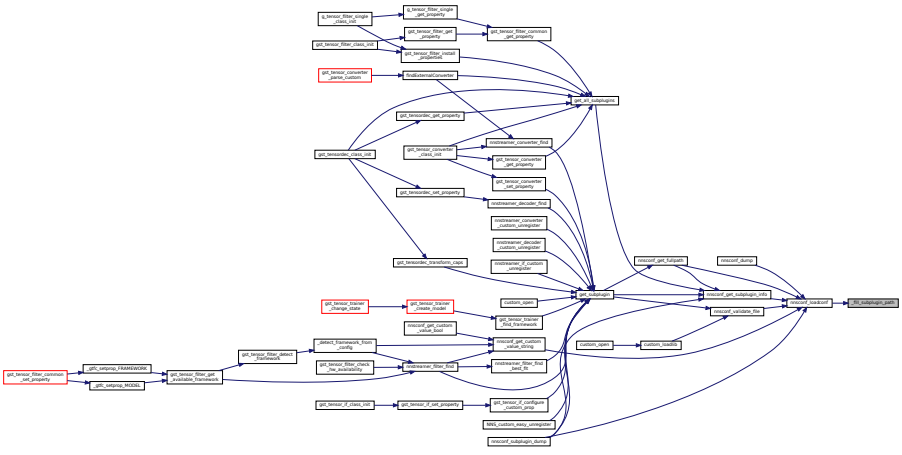
9.88.4.2 _fill_subplugin_path()

```
static void _fill_subplugin_path (
            confdata * cdata,
            GKeyFile * key_file,
            conf_sources src )  [static]
```

Private function to fill subplugin path.

Definition at line 326 of file nnstreamer_conf.c.

Here is the caller graph for this function:



9.88.4.3 _foreach_custom_property()

```
static void _foreach_custom_property (
             GQuark key_id,
             gpointer data,
             gpointer user_data )  [static]
```

foreach callback for custom property

Definition at line 681 of file nnstreamer_conf.c.

Here is the caller graph for this function:

9.88.4.4 _g_list_foreach_vstr_helper()

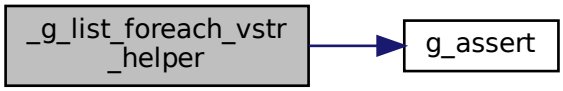
```
static void _g_list_foreach_vstr_helper (
             gpointer data,
             gpointer user_data )  [static]
```

Private function to help convert linked-list to vstr with foreach @data The element data of linked-list @user_data The struct to fill in vstr.

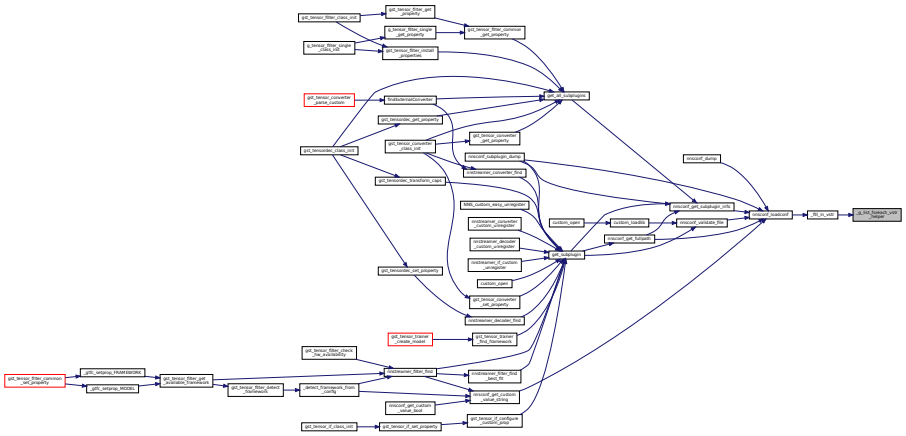
library error? internal logic error?

Definition at line 268 of file nnstreamer_conf.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.88.4.5 _get_filenames()

```
static gboolean _get_filenames (
            nnsconf_type_path type,
            const gchar * dir,
            GSList ** listF,
            GSList ** listN,
            guint * counter )  [static]
```

Private function to fill in ".so/.dylib list" with fullpath-filenames in a directory.

Parameters

in	*type*	conf type to scan.
in	*dir*	Directory to be searched.
	[in/out]	listF The fullpath list to be updated.
	[in/out]	listN The name list to be updated.
	[in/out]	counter increased by the number of appended elements.

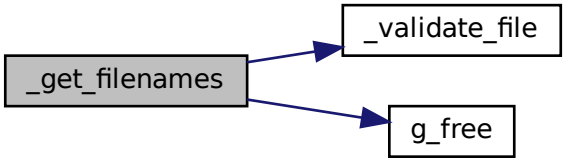
Returns

True if successfully updated.

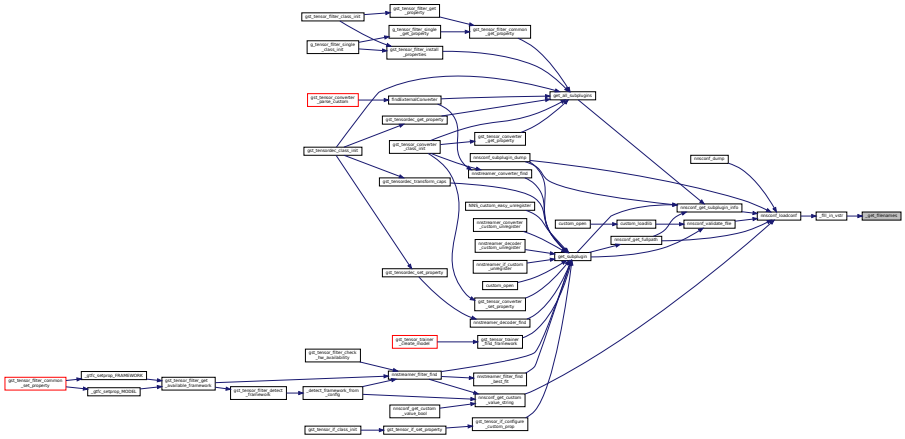
Todo This assumes .so/.dylib for all sub plugins. Support Windows!

Definition at line 183 of file nnstreamer_conf.c.

Here is the call graph for this function:



Here is the caller graph for this function:



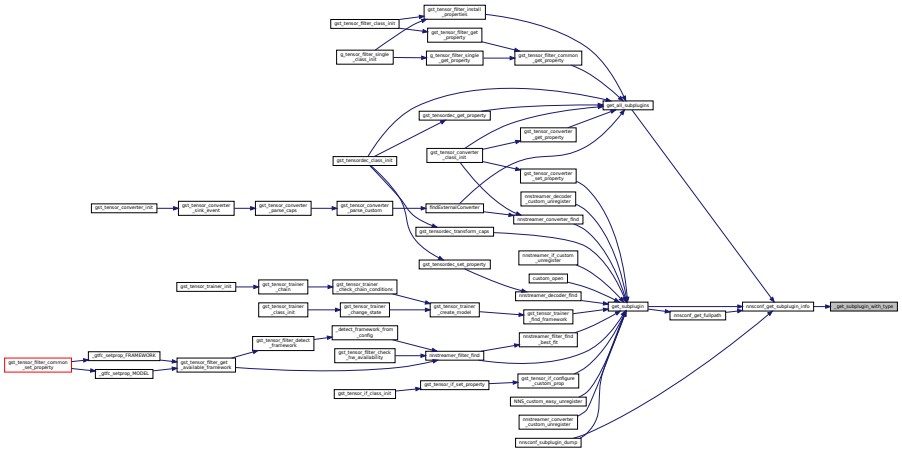
9.88.4.6 _get_subplugin_with_type()

```
static gboolean _get_subplugin_with_type (
            nnsconf_type_path type,
            gchar *** name,
            gchar *** filepath )  [static]
```

Private function to get sub-plugins list with type.

Definition at line 229 of file nnstreamer_conf.c.

Here is the caller graph for this function:



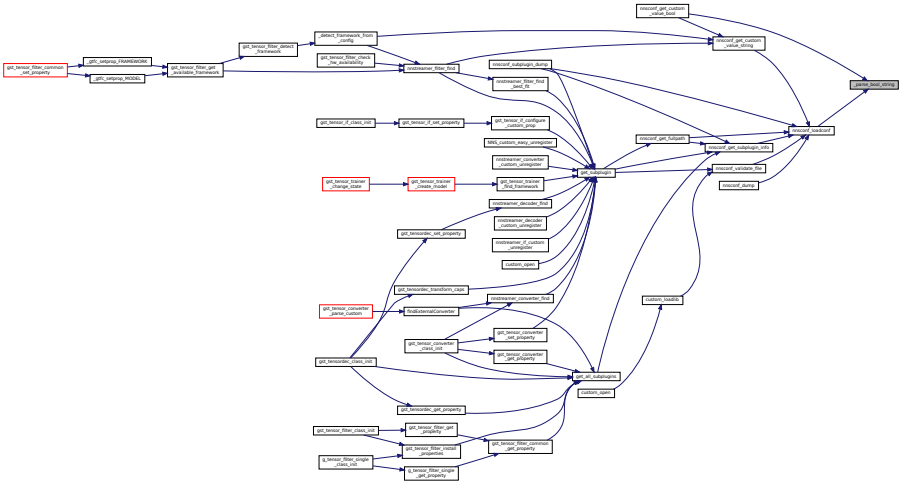
9.88.4.7 _parse_bool_string()

```
static gboolean _parse_bool_string (
            const gchar * strval,
            gboolean def )  [static]
```

Parse string to get boolean value.

Definition at line 115 of file nnstreamer_conf.c.

Here is the caller graph for this function:



9.88.4.8 _strdup_getenv()

```
static gchar* _strdup_getenv (
            const gchar * name )  [static]
```

Private function to get strdup-ed env-var if it's valid. Otherwise, NULL.

Return values

strdup-ed	env-var value

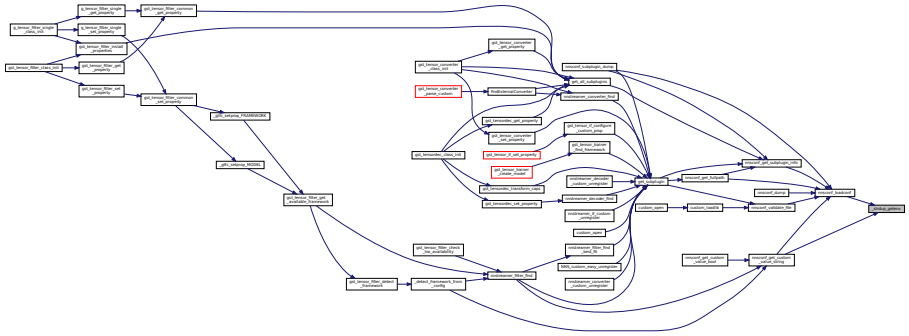
Parameters

in	*name*	Environmental variable name

Todo Evaluate if we need to use secure_getenv() here (and compatible with other OS

Definition at line 143 of file nnstreamer_conf.c.

Here is the caller graph for this function:



9.88.4.9 _validate_file()

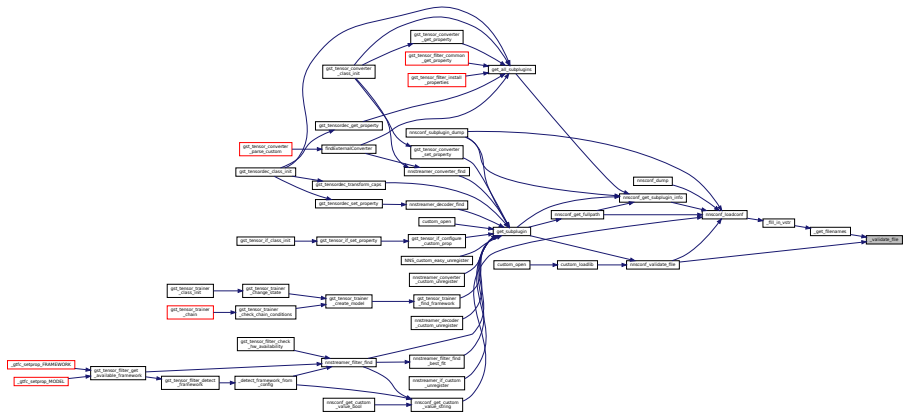
```
static gboolean _validate_file (
            nnsconf_type_path type,
            const gchar * fullpath )  [static]
```

Private function to validate .so file can be added to the list.

Todo how to validate with nnsconf type.

Definition at line 158 of file nnstreamer_conf.c.

Here is the caller graph for this function:



9.88.4.10 nnsconf_dump()

```
void nnsconf_dump (
            gchar * str,
            gulong size )
```

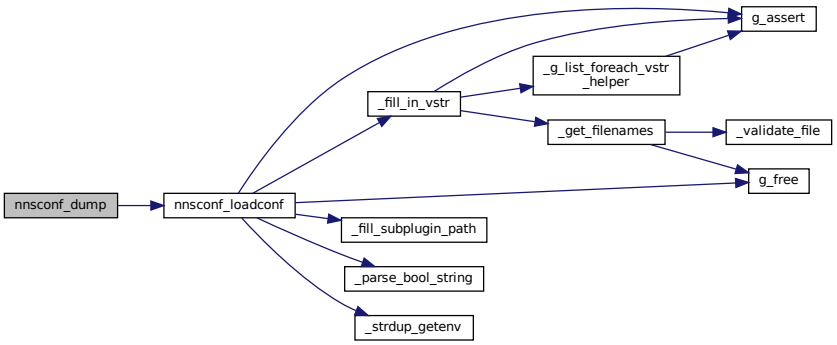
Print out configurations.

NNStreamer configuration dump as string.

Todo Add more configuration values to dump.

Definition at line 628 of file nnstreamer_conf.c.

Here is the call graph for this function:



9.88.4.11 nnsconf_get_custom_value_bool()

```
gboolean nnsconf_get_custom_value_bool (
            const gchar * group,
            const gchar * key,
            gboolean def )
```

Public function defined in the header.

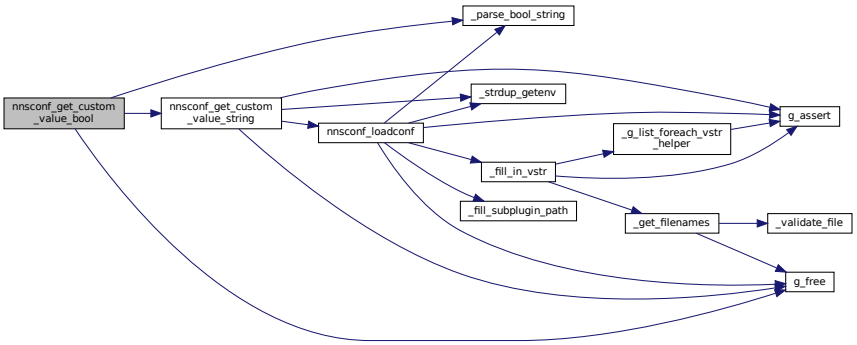
Get the custom configuration value from .ini and envvar. @detail For predefined configurations defined in this header, use the given enum for faster configuration processing. For custom configurations not defined in this header, you may use this API to access your own custom configurations. Configuration values may be loaded only once during runtime, thus, if the values are changed in run-time, the changes are not guaranteed to be reflected. The ENVVAR is supposed to be NNSTREAMER_${group}_${key}, which has higher priority than the .ini configuration. Be careful not to use special characters in group name ([,], _).

Note

This function is included in nnstreamer internal header for native APIs. When changing the declaration, you should update the internal header (nnstreamer_internal.h).

Definition at line 609 of file nnstreamer_conf.c.

Here is the call graph for this function:



9.88.4.12 nnsconf_get_custom_value_string()

```
gchar* nnsconf_get_custom_value_string (
            const gchar * group,
            const gchar * key )
```

Public function defined in the header.

Get the custom configuration value from .ini and envvar. @detail For predefined configurations defined in this header, use the given enum for faster configuration processing. For custom configurations not defined in this header, you may use this API to access your own custom configurations. Configuration values may be loaded only once during runtime, thus, if the values are changed in run-time, the changes are not guaranteed to be reflected. The ENVVAR is supposed to be NNSTREAMER_${group}_${key}, which has higher priority than the .ini configuration. Be careful not to use special characters in group name ([,], _).

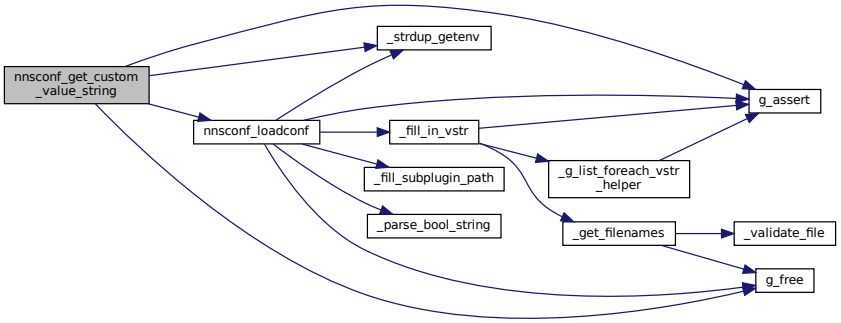
Note

This function is included in nnstreamer internal header for native APIs. When changing the declaration, you should update the internal header (nnstreamer_internal.h).

Internal lib error? out-of-memory?

Definition at line 557 of file nnstreamer_conf.c.

Here is the call graph for this function:



Here is the caller graph for this function:

9.88.4.13 nnsconf_get_fullpath()

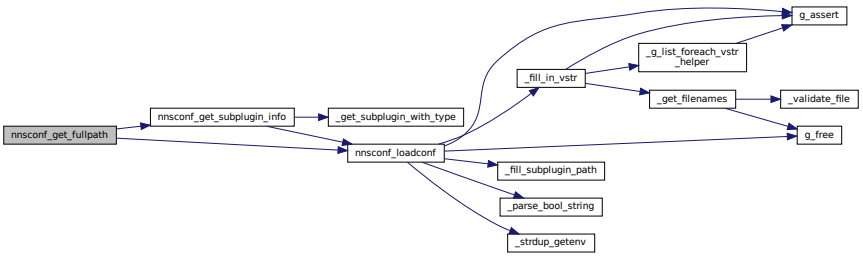
```
const gchar* nnsconf_get_fullpath (
            const gchar * subpluginname,
            nnsconf_type_path type )
```

Public function defined in the header.

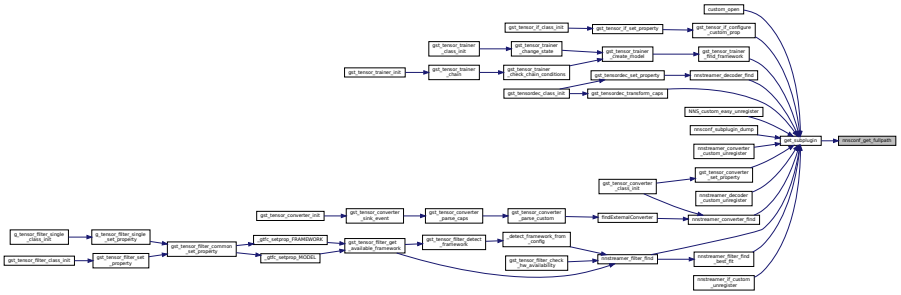
Get the configured paths for the type with sub-plugin name.

Definition at line 483 of file nnstreamer_conf.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.88.4.14 nnsconf_get_subplugin_info()

```
guint nnsconf_get_subplugin_info (
            nnsconf_type_path type,
            subplugin_info_s * info )
```

Public function to get the list of sub-plugins name and path.

Returns

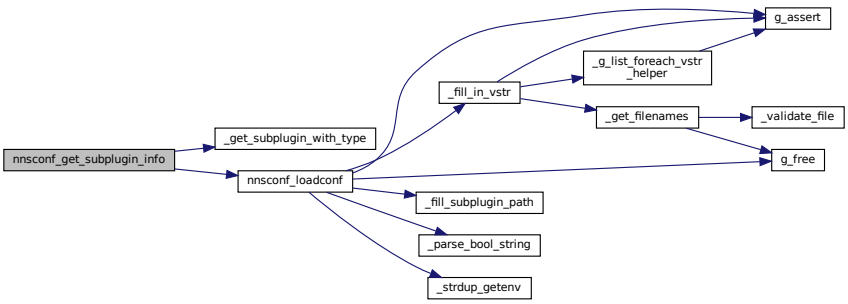
total number of sub-plugins for given type

Note

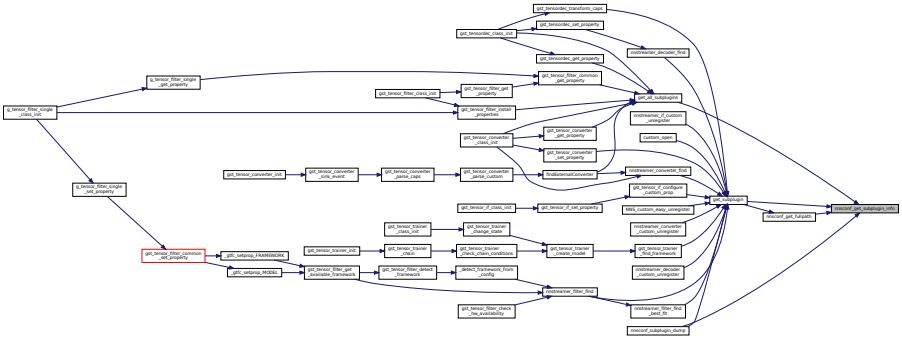
DO NOT free sub-plugins info

Definition at line 528 of file nnstreamer_conf.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.88.4.15 nnsconf_get_subplugin_name_prefix()

```
const gchar* nnsconf_get_subplugin_name_prefix (
            nnsconf_type_path type )
```

Get sub-plugin's name prefix.

Parameters

in	*type*	The type (FILTERS/DECODERS/CUSTOM_FILTERS)

Returns

Predefined prefix string for given type.

Definition at line 516 of file nnstreamer_conf.c.

9.88.4.16 nnsconf_loadconf()

```
gboolean nnsconf_loadconf (
            gboolean force_reload )
```

Public function defined in the header.

Load the .ini file. if it's not Tizen, configuration from env-var has a higher priority

Priority of reading a conf file 1) read from NNSTREAMER_CONF_FILE 2) read from NNSTREAMER_DEFAULT↩
_CONF_FILE 3) read from env-var

default value of 'sysconfdir' in meson is 'etc'

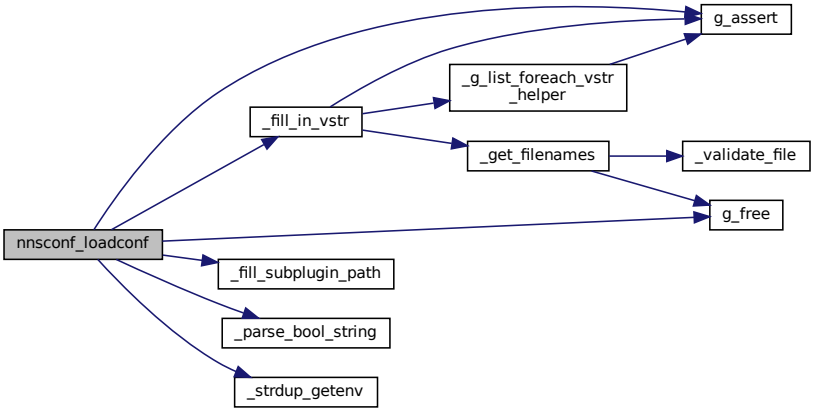
Internal lib error? out-of-memory?

Internal lib error? out-of-memory?

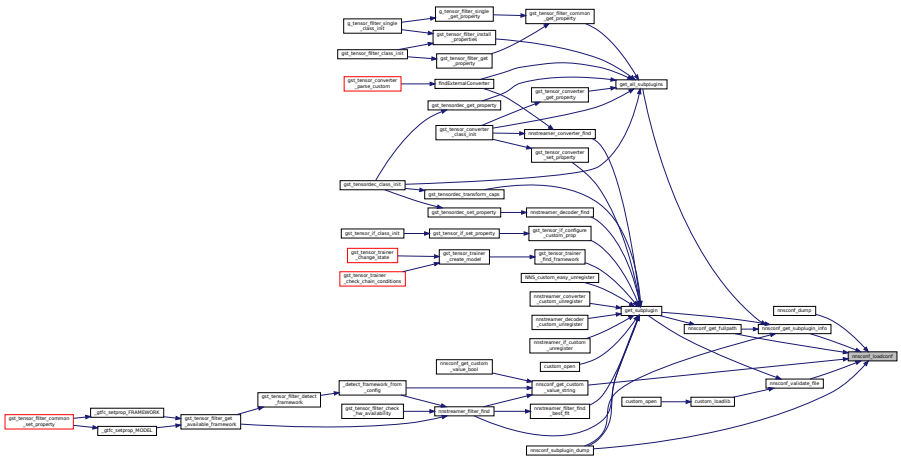
Failed to get the configuration. Note that Android API does not use the configuration.

Definition at line 342 of file nnstreamer_conf.c.

Here is the call graph for this function:



Here is the caller graph for this function:



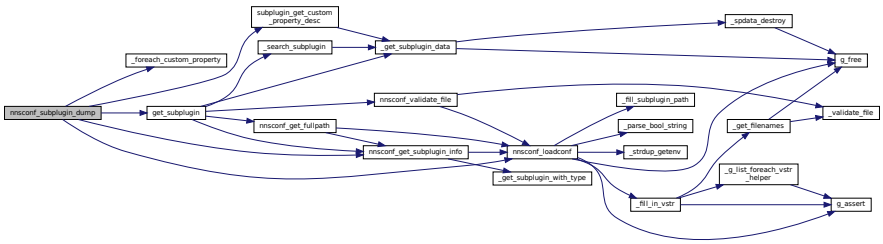
9.88.4.17 nnsconf_subplugin_dump()

```
void nnsconf_subplugin_dump (
            gchar * str,
            gulong size )
```

Print out the information of registered sub-plugins.

Definition at line 695 of file nnstreamer_conf.c.

Here is the call graph for this function:



9.88.4.18 nnsconf_validate_file()

```
gboolean nnsconf_validate_file (
            nnsconf_type_path type,
            const gchar * fullpath )
```

Public function to validate sub-plugin library is available.

Parameters

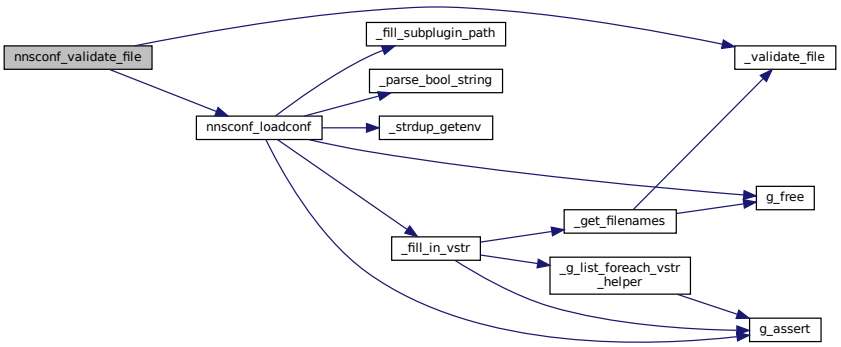
in	*type*	The type (FILTERS/DECODERS/CUSTOM_FILTERS)
in	*fullpath*	The full path to the file.

Returns

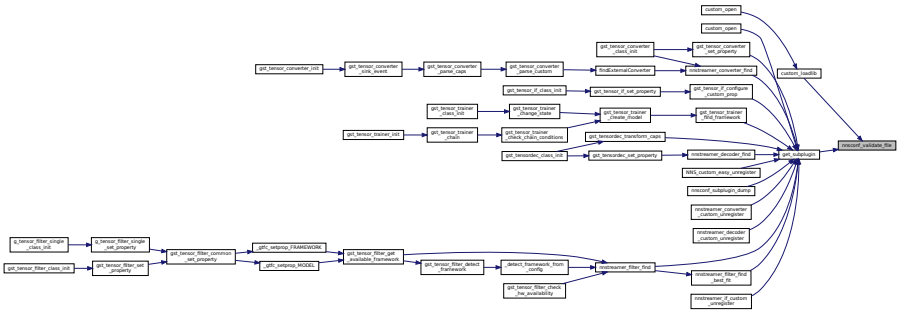
True if the file is regular and can be added to the list.

Definition at line 503 of file nnstreamer_conf.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.88.5 Variable Documentation

9.88.5.1 conf

```
confdata conf = { 0 }  [static]
```

Definition at line 109 of file nnstreamer_conf.c.

9.88.5.2 custom_table

```
GHashTable* custom_table = NULL  [static]
```

Internal cache for the custom key-values.

Definition at line 549 of file nnstreamer_conf.c.

9.88.5.3 NNSTREAMER_ENVVAR

```
const gchar* NNSTREAMER_ENVVAR[NNSCONF_PATH_END]  [static]
```

Initial value:

```
= {
  [NNSCONF_PATH_FILTERS] = "NNSTREAMER_FILTERS",
  [NNSCONF_PATH_DECODERS] = "NNSTREAMER_DECODERS",
  [NNSCONF_PATH_CUSTOM_FILTERS] = "NNSTREAMER_CUSTOMFILTERS",
  [NNSCONF_PATH_CONVERTERS] = "NNSTREAMER_CONVERTERS",
  [NNSCONF_PATH_TRAINERS] = "NNSTREAMER_TRAINERS"
}
```

Definition at line 47 of file nnstreamer_conf.c.

9.88.5.4 NNSTREAMER_PATH

```
const gchar* NNSTREAMER_PATH[NNSCONF_PATH_END]  [static]
```

Initial value:

```
= {
  [NNSCONF_PATH_FILTERS] = "/usr/lib/nnstreamer/filters/",
  [NNSCONF_PATH_DECODERS] = "/usr/lib/nnstreamer/decoders/",
  [NNSCONF_PATH_CUSTOM_FILTERS] = "/usr/lib/nnstreamer/customfilters/",
  [NNSCONF_PATH_CONVERTERS] = "/usr/lib/nnstreamer/converters/",
  [NNSCONF_PATH_TRAINERS] = "/usr/lib/nnstreamer/trainers/"
}
```

Definition at line 55 of file nnstreamer_conf.c.

9.88.5.5 subplugin_prefixes

```
const gchar* subplugin_prefixes[]  [static]
```

Initial value:

```
= {
  [NNSCONF_PATH_FILTERS] = NNSTREAMER_PREFIX_FILTER,
  [NNSCONF_PATH_DECODERS] = NNSTREAMER_PREFIX_DECODER,
  [NNSCONF_PATH_CUSTOM_FILTERS] = NNSTREAMER_PREFIX_CUSTOMFILTERS,
  [NNSCONF_PATH_EASY_CUSTOM_FILTERS] = NNSTREAMER_PREFIX_CUSTOMFILTERS,
  [NNSCONF_PATH_CONVERTERS] = NNSTREAMER_PREFIX_CONVERTER,
  [NNSCONF_PATH_TRAINERS] = NNSTREAMER_PREFIX_TRAINER,
  [NNSCONF_PATH_END] = NULL
}
```

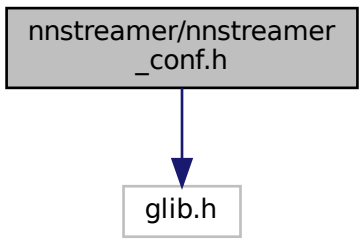
Definition at line 63 of file nnstreamer_conf.c.

9.89 nnstreamer/nnstreamer_conf.h File Reference

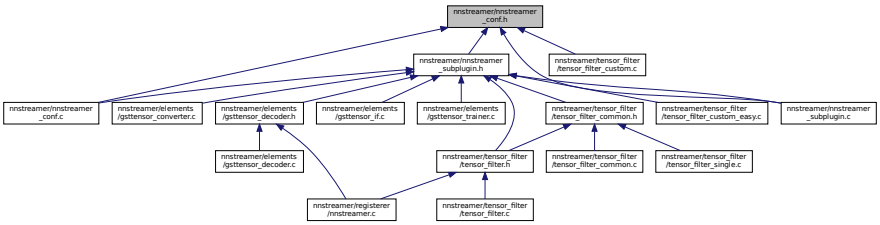
Internal header for conf/env-var management.

```
#include <glib.h>
```

Include dependency graph for nnstreamer_conf.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct subplugin_info_s

Macros

- #define NNSTREAMER_SYS_ROOT_PATH_PREFIX "/"
- #define NNSTREAMER_SO_FILE_EXTENSION ".so"
- #define NNSTREAMER_DEFAULT_CONF_FILE "/etc/nnstreamer.ini"
- #define NNSTREAMER_CONF_FILE NNSTREAMER_DEFAULT_CONF_FILE
- #define NNSTREAMER_ENVVAR_CONF_FILE "NNSTREAMER_CONF"

Enumerations

- enum nnsconf_type_path {
  NNSCONF_PATH_FILTERS = 0, NNSCONF_PATH_DECODERS, NNSCONF_PATH_CUSTOM_FILTERS,
  NNSCONF_PATH_EASY_CUSTOM_FILTERS,
  NNSCONF_PATH_CONVERTERS, NNSCONF_PATH_TRAINERS, NNSCONF_PATH_END }

Functions

- gboolean nnsconf_loadconf (gboolean force_reload)

  Load the .ini file.
- const gchar * nnsconf_get_fullpath (const gchar *subpluginname, nnsconf_type_path type)

  Get the configured paths for the type with sub-plugin name.
- gboolean nnsconf_validate_file (nnsconf_type_path type, const gchar *fullpath)

  Public function to validate sub-plugin library is available.
- const gchar * nnsconf_get_subplugin_name_prefix (nnsconf_type_path type)

  Get sub-plugin's name prefix.
- guint nnsconf_get_subplugin_info (nnsconf_type_path type, subplugin_info_s *info)

  Public function to get the list of sub-plugins name and path.
- gchar * nnsconf_get_custom_value_string (const gchar *group, const gchar *key)

  Get the custom configuration value from .ini and envvar. @detail For predefined configurations defined in this header, use the given enum for faster configuration processing. For custom configurations not defined in this header, you may use this API to access your own custom configurations. Configuration values may be loaded only once during runtime, thus, if the values are changed in run-time, the changes are not guaranteed to be reflected. The ENVVAR is supposed to be NNSTREAMER_${group}_${key}, which has higher priority than the .ini configuration. Be careful not to use special characters in group name ([,], _).
- gboolean nnsconf_get_custom_value_bool (const gchar *group, const gchar *key, gboolean def)

Get the custom configuration value from .ini and envvar. @detail For predefined configurations defined in this header, use the given enum for faster configuration processing. For custom configurations not defined in this header, you may use this API to access your own custom configurations. Configuration values may be loaded only once during runtime, thus, if the values are changed in run-time, the changes are not guaranteed to be reflected. The ENVVAR is supposed to be NNSTREAMER_${group}_${key}, which has higher priority than the .ini configuration. Be careful not to use special characters in group name ([,], _).

- void nnsconf_dump (gchar *str, gulong size)

  NNStreamer configuration dump as string.
- void nnsconf_subplugin_dump (gchar *str, gulong size)

  Print out the information of registered sub-plugins.

9.89.1 Detailed Description

Internal header for conf/env-var management.

NNStreamer Configurations / Environmental Variable Manager. Copyright (C) 2018 MyungJoo Ham myungjoo.ham@samsung.com

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Library General Public License as published by the Free Software Foundation; version 2.1 of the License.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Library General Public License for more details.

Date

26 Nov 2018

See also

https://github.com/nnstreamer/nnstreamer

Author

MyungJoo Ham myungjoo.ham@samsung.com

Bug No known bugs except for NYI items

If there are duplicated configurations for the same element, the one with higher priority may override (if it cannot be stacked up),

- (Highest) Environmental variables
- The configuration file (default: /etc/nnstreamer.ini)
- (Lowest) Internal hardcoded values.

Do not export this to devel package. This is an internal header.

9.89.2 Macro Definition Documentation

9.89.2.1 NNSTREAMER_CONF_FILE

```
#define NNSTREAMER_CONF_FILE NNSTREAMER_DEFAULT_CONF_FILE
```

Definition at line 59 of file nnstreamer_conf.h.

9.89.2.2 NNSTREAMER_DEFAULT_CONF_FILE

```
#define NNSTREAMER_DEFAULT_CONF_FILE "/etc/nnstreamer.ini"
```

Definition at line 57 of file nnstreamer_conf.h.

9.89.2.3 NNSTREAMER_ENVVAR_CONF_FILE

```
#define NNSTREAMER_ENVVAR_CONF_FILE "NNSTREAMER_CONF"
```

Definition at line 61 of file nnstreamer_conf.h.

9.89.2.4 NNSTREAMER_SO_FILE_EXTENSION

```
#define NNSTREAMER_SO_FILE_EXTENSION ".so"
```

Hard-coded system-dependent file extension string of shared (dynamic loadable) object

Definition at line 53 of file nnstreamer_conf.h.

9.89.2.5 NNSTREAMER_SYS_ROOT_PATH_PREFIX

```
#define NNSTREAMER_SYS_ROOT_PATH_PREFIX "/"
```

Definition at line 43 of file nnstreamer_conf.h.

9.89.3 Enumeration Type Documentation

9.89.3.1 nnsconf_type_path

```
enum nnsconf_type_path
```

Enumerator

NNSCONF_PATH_FILTERS	
NNSCONF_PATH_DECODERS	
NNSCONF_PATH_CUSTOM_FILTERS	
NNSCONF_PATH_EASY_CUSTOM_FILTERS	
NNSCONF_PATH_CONVERTERS	
NNSCONF_PATH_TRAINERS	
NNSCONF_PATH_END	

Definition at line 63 of file nnstreamer_conf.h.

9.89.4 Function Documentation

9.89.4.1 nnsconf_dump()

```
void nnsconf_dump (
            gchar * str,
            gulong size )
```

NNStreamer configuration dump as string.

Parameters

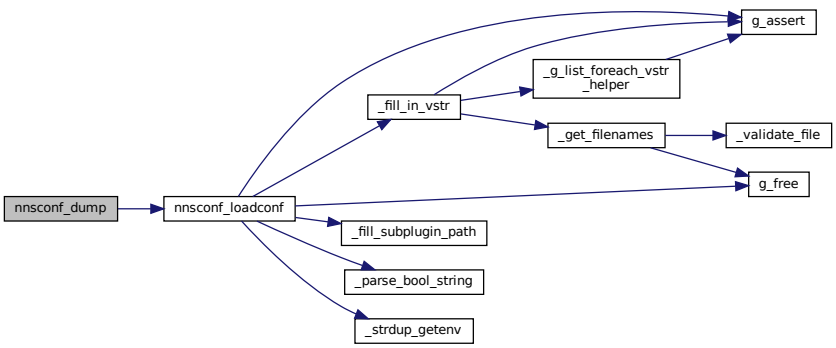
out	*str*	Preallocated string for the output (dump).
in	*size*	The size of given str.

NNStreamer configuration dump as string.

Todo Add more configuration values to dump.

Definition at line 628 of file nnstreamer_conf.c.

Here is the call graph for this function:



9.89.4.2 nnsconf_get_custom_value_bool()

```
gboolean nnsconf_get_custom_value_bool (
            const gchar * group,
            const gchar * key,
            gboolean def )
```

Get the custom configuration value from .ini and envvar. @detail For predefined configurations defined in this header, use the given enum for faster configuration processing. For custom configurations not defined in this header, you may use this API to access your own custom configurations. Configuration values may be loaded only once during runtime, thus, if the values are changed in run-time, the changes are not guaranteed to be reflected. The ENVVAR is supposed to be NNSTREAMER_${group}_${key}, which has higher priority than the .ini configuration. Be careful not to use special characters in group name ([,], _).

Parameters

in	*group*	The group name, [group], in .ini file.
in	*key*	The key name, key = value, in .ini file.
in	*def*	The default return value in case there is no value available.

Returns

The value interpreted as TRUE/FALSE.

Get the custom configuration value from .ini and envvar. @detail For predefined configurations defined in this header, use the given enum for faster configuration processing. For custom configurations not defined in this header, you may use this API to access your own custom configurations. Configuration values may be loaded only once during runtime, thus, if the values are changed in run-time, the changes are not guaranteed to be reflected. The ENVVAR is supposed to be NNSTREAMER_${group}_${key}, which has higher priority than the .ini configuration. Be careful not to use special characters in group name ([,], _).

Note

This function is included in nnstreamer internal header for native APIs. When changing the declaration, you should update the internal header (nnstreamer_internal.h).

Definition at line 609 of file nnstreamer_conf.c.

9.89.4.3 nnsconf_get_custom_value_string()

```
gchar* nnsconf_get_custom_value_string (
            const gchar * group,
            const gchar * key )
```

Get the custom configuration value from .ini and envvar. @detail For predefined configurations defined in this header, use the given enum for faster configuration processing. For custom configurations not defined in this header, you may use this API to access your own custom configurations. Configuration values may be loaded only once during runtime, thus, if the values are changed in run-time, the changes are not guaranteed to be reflected. The ENVVAR is supposed to be NNSTREAMER_${group}_${key}, which has higher priority than the .ini configuration. Be careful not to use special characters in group name ([,], _).

Parameters

in	*group*	The group name, [group], in .ini file.
in	*key*	The key name, key = value, in .ini file.

Returns

The newly allocated string. A caller must free it. NULL if it's not available.

Get the custom configuration value from .ini and envvar. @detail For predefined configurations defined in this header, use the given enum for faster configuration processing. For custom configurations not defined in this header, you may use this API to access your own custom configurations. Configuration values may be loaded only once during runtime, thus, if the values are changed in run-time, the changes are not guaranteed to be reflected. The ENVVAR is supposed to be NNSTREAMER_${group}_${key}, which has higher priority than the .ini configuration. Be careful not to use special characters in group name ([,], _).

Note

This function is included in nnstreamer internal header for native APIs. When changing the declaration, you should update the internal header (nnstreamer_internal.h).

Internal lib error? out-of-memory?

Definition at line 557 of file nnstreamer_conf.c.

9.89.4.4 nnsconf_get_fullpath()

```
const gchar* nnsconf_get_fullpath (
             const gchar * subpluginname,
             nnsconf_type_path type )
```

Get the configured paths for the type with sub-plugin name.

Parameters

in	*The*	subplugin name except for the prefix and postfix (.so) to find
in	*type*	The type (FILTERS/DECODERS/CUSTOM_FILTERS)

Returns

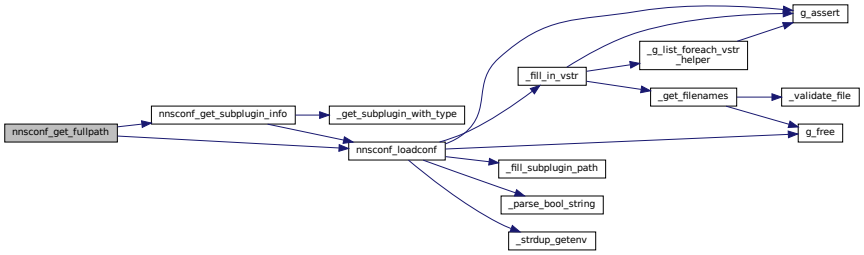
The full path to the file. Caller MUST NOT modify this. Returns NULL if we cannot find the file.

This is mainly supposed to be used by CUSTOM_FILTERS

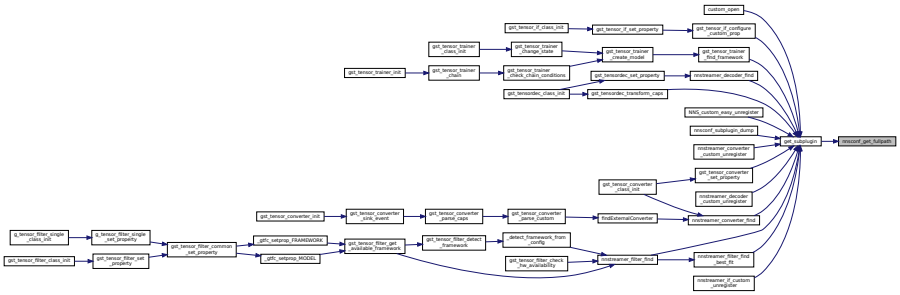
Get the configured paths for the type with sub-plugin name.

Definition at line 483 of file nnstreamer_conf.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.89.4.5 nnsconf_get_subplugin_info()

```
guint nnsconf_get_subplugin_info (
            nnsconf_type_path type,
            subplugin_info_s * info )
```

Public function to get the list of sub-plugins name and path.

Parameters

in	*type*	The type (FILTERS/DECODERS/CUSTOM_FILTERS)
out	*info*	The data structure which contains the name and full path of sub-plugins

Returns

total number of sub-plugins for given type

Note

DO NOT free sub-plugins info

Returns

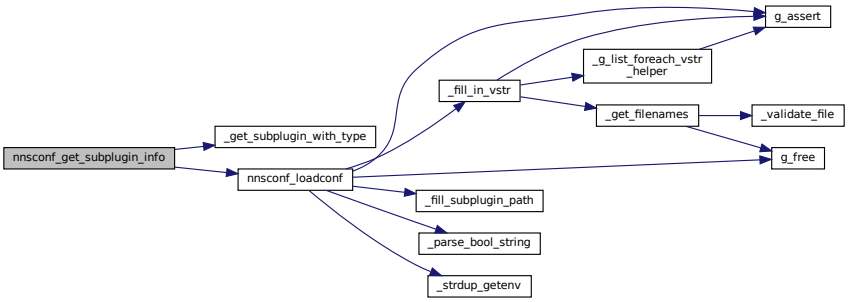
total number of sub-plugins for given type

Note

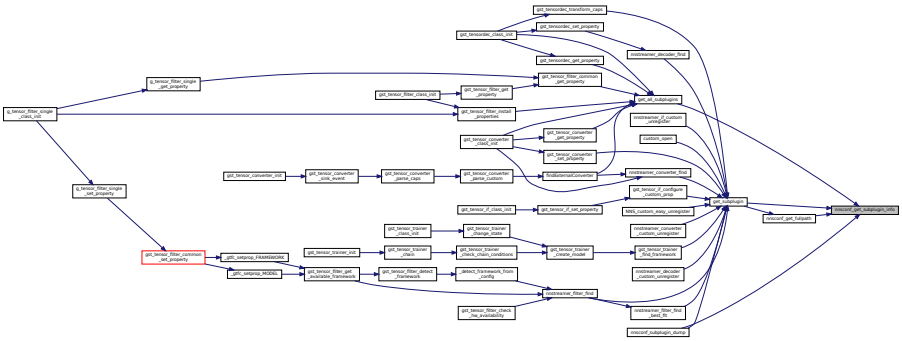
DO NOT free sub-plugins info

Definition at line 528 of file nnstreamer_conf.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.89.4.6 nnsconf_get_subplugin_name_prefix()

```
const gchar* nnsconf_get_subplugin_name_prefix (
            nnsconf_type_path type )
```

Get sub-plugin's name prefix.

Parameters

in	*type*	The type (FILTERS/DECODERS/CUSTOM_FILTERS)

Returns

Predefined prefix string for given type.

Definition at line 516 of file nnstreamer_conf.c.

9.89.4.7 nnsconf_loadconf()

```
gboolean nnsconf_loadconf (
            gboolean force_reload )
```

Load the .ini file.

Parameters

in	*force_reload*	TRUE if you want to clean up and load conf again.

Returns

TRUE if successful or skipped. FALSE if error reading something.

Load the .ini file. if it's not Tizen, configuration from env-var has a higher priority

Priority of reading a conf file 1) read from NNSTREAMER_CONF_FILE 2) read from NNSTREAMER_DEFAULT↩ _CONF_FILE 3) read from env-var

default value of 'sysconfdir' in meson is 'etc'

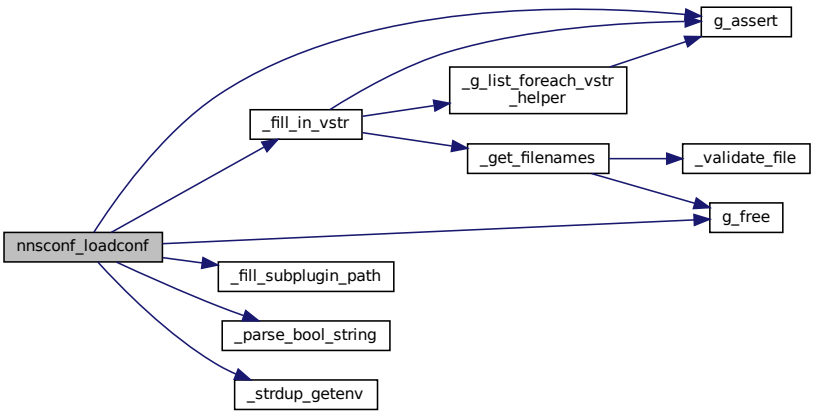
Internal lib error? out-of-memory?

Internal lib error? out-of-memory?

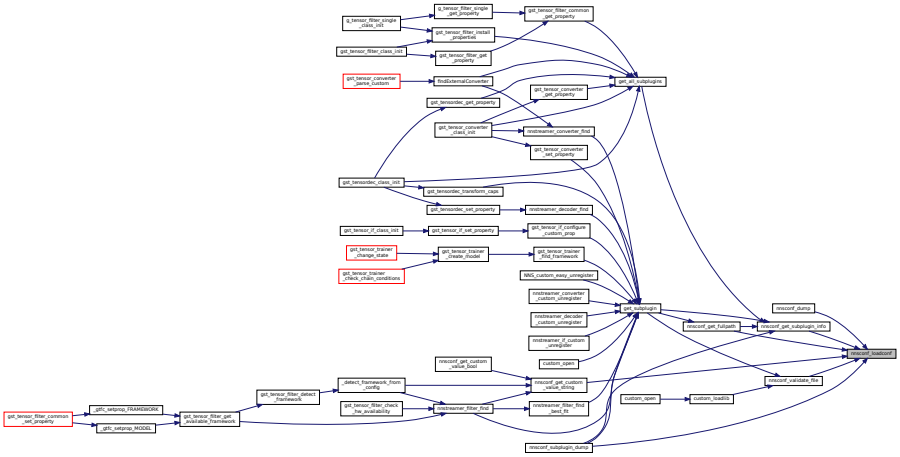
Failed to get the configuration. Note that Android API does not use the configuration.

Definition at line 342 of file nnstreamer_conf.c.

Here is the call graph for this function:



Here is the caller graph for this function:



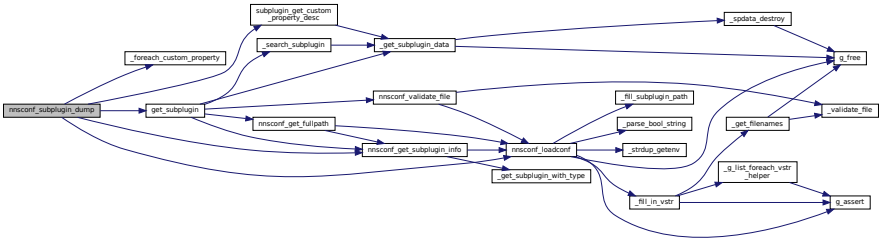
9.89.4.8 nnsconf_subplugin_dump()

```
void nnsconf_subplugin_dump (
            gchar * str,
            gulong size )
```

Print out the information of registered sub-plugins.

Definition at line 695 of file nnstreamer_conf.c.

Here is the call graph for this function:



9.89.4.9 nnsconf_validate_file()

```
gboolean nnsconf_validate_file (
            nnsconf_type_path type,
            const gchar * fullpath )
```

Public function to validate sub-plugin library is available.

Parameters

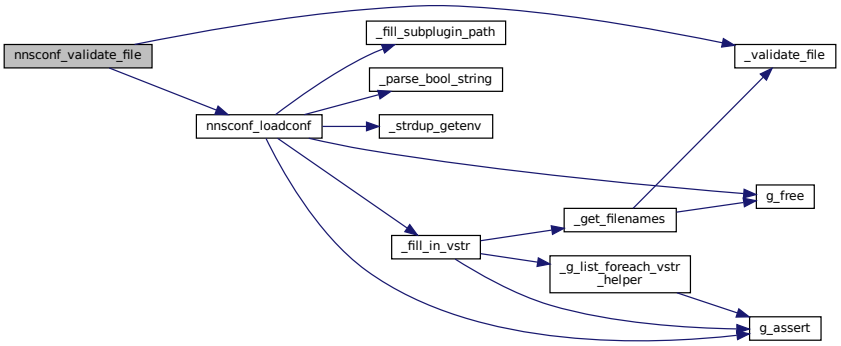
in	*type*	The type (FILTERS/DECODERS/CUSTOM_FILTERS)
in	*fullpath*	The full path to the file.

Returns

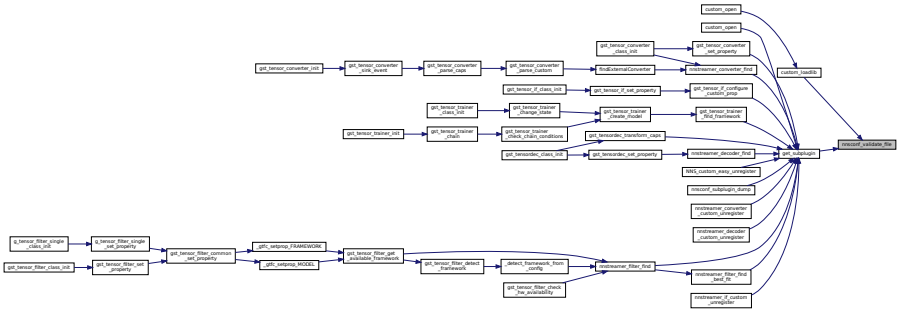
True if the file is regular and can be added to the list.

Definition at line 503 of file nnstreamer_conf.c.

Here is the call graph for this function:



Here is the caller graph for this function:

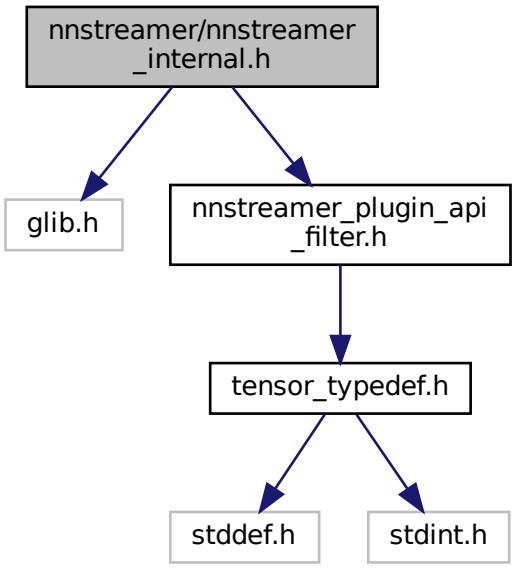


9.90 nnstreamer/nnstreamer_internal.h File Reference

Internal header for NNStreamer plugins and native single-shot APIs.

```
#include <glib.h>
#include <nnstreamer_plugin_api_filter.h>
```

Include dependency graph for nnstreamer_internal.h:



Functions

- G_BEGIN_DECLS gchar * nnsconf_get_custom_value_string (const gchar *group, const gchar *key)

  Get the custom configuration value from .ini and envvar. @detail For predefined configurations defined in this header, use the given enum for faster configuration processing. For custom configurations not defined in this header, you may use this API to access your own custom configurations. Configuration values may be loaded only once during runtime, thus, if the values are changed in run-time, the changes are not guaranteed to be reflected. The ENVVAR is supposed to be NNSTREAMER_${group}_${key}, which has higher priority than the .ini configuration. Be careful not to use special characters in group name ([,], _).

- gboolean nnsconf_get_custom_value_bool (const gchar *group, const gchar *key, gboolean def)

  Get the custom configuration value from .ini and envvar. @detail For predefined configurations defined in this header, use the given enum for faster configuration processing. For custom configurations not defined in this header, you may use this API to access your own custom configurations. Configuration values may be loaded only once during runtime, thus, if the values are changed in run-time, the changes are not guaranteed to be reflected. The ENVVAR is supposed to be NNSTREAMER_${group}_${key}, which has higher priority than the .ini configuration. Be careful not to use special characters in group name ([,], _).

- gchar * gst_tensor_filter_detect_framework (const gchar *const *model_files, const guint num_models, const gboolean load_conf)

  Get neural network framework name from given model file. This does not guarantee the framework is available on the target device.

- gboolean gst_tensor_filter_check_hw_availability (const gchar *name, const accl_hw hw, const char *custom)

  Check if the given hw is supported by the framework.

9.90.1 Detailed Description

Internal header for NNStreamer plugins and native single-shot APIs.

Date

28 Jan 2021

See also

http://github.com/nnstreamer/nnstreamer

Author

Jaeyun Jung jy1210.jung@samsung.com

Bug No known bugs except for NYI items

9.90.2 Function Documentation

9.90.2.1 gst_tensor_filter_check_hw_availability()

```
gboolean gst_tensor_filter_check_hw_availability (
            const gchar * name,
            const accl_hw hw,
            const char * custom )
```

Check if the given hw is supported by the framework.

Parameters

in	*name*	The name of filter sub-plugin.
in	*hw*	Backend accelerator hardware.
in	*custom*	User-defined string to handle detailed hardware option.

Returns

TRUE if given hw is available.

Note

This function is included in nnstreamer internal header for native APIs. When changing the declaration, you should update the internal header (nnstreamer_internal.h).

Only check for specific HW, DEFAULT/AUTO are always supported

Definition at line 2923 of file tensor_filter_common.c.

9.90.2.2 gst_tensor_filter_detect_framework()

```
gchar* gst_tensor_filter_detect_framework (
            const gchar *const * model_files,
            const guint num_models,
            const gboolean load_conf )
```

Get neural network framework name from given model file. This does not guarantee the framework is available on the target device.

Parameters

in	*model_files*	the prediction model paths
in	*num_models*	the number of model files
in	*load_conf*	flag to load configuration for the priority of framework

Returns

Possible framework name (NULL if it fails to detect automatically). Caller should free returned value using g_free().

Parameters

in	*model_files*	the prediction model paths
in	*num_models*	the number of model files
in	*load_conf*	flag to load configuration for the priority of framework

Returns

Possible framework name (NULL if it fails to detect automatically). Caller should free returned value using g_free().

Note

This function is included in nnstreamer internal header for native APIs. When changing the declaration, you should update the internal header (nnstreamer_internal.h).

Note

When adding new file extension for auto fw detection, you should check the condition to validate model file - ml_validate_model_file() in ML-API.

Definition at line 1232 of file tensor_filter_common.c.

9.90.2.3 nnsconf_get_custom_value_bool()

```
gboolean nnsconf_get_custom_value_bool (
            const gchar * group,
            const gchar * key,
            gboolean def )
```

Get the custom configuration value from .ini and envvar. @detail For predefined configurations defined in this header, use the given enum for faster configuration processing. For custom configurations not defined in this header, you may use this API to access your own custom configurations. Configuration values may be loaded only once during runtime, thus, if the values are changed in run-time, the changes are not guaranteed to be reflected. The ENVVAR is supposed to be NNSTREAMER_${group}_${key}, which has higher priority than the .ini configuration. Be careful not to use special characters in group name ([,], _).

Parameters

in	*group*	The group name, [group], in .ini file.
in	*key*	The key name, key = value, in .ini file.
in	*def*	The default return value in case there is no value available.

Returns

The value interpreted as TRUE/FALSE.

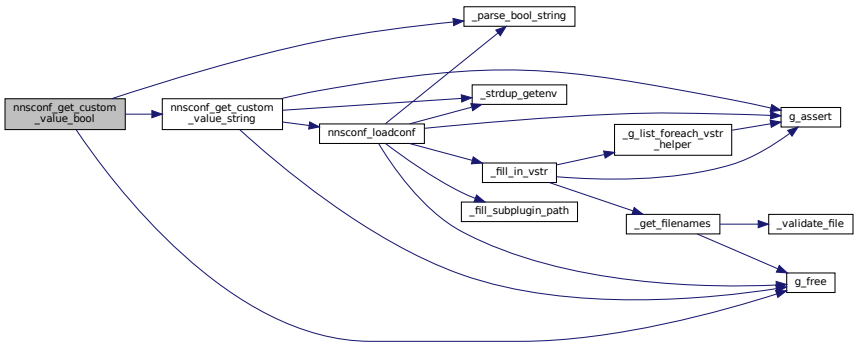
Get the custom configuration value from .ini and envvar. @detail For predefined configurations defined in this header, use the given enum for faster configuration processing. For custom configurations not defined in this header, you may use this API to access your own custom configurations. Configuration values may be loaded only once during runtime, thus, if the values are changed in run-time, the changes are not guaranteed to be reflected. The ENVVAR is supposed to be NNSTREAMER_${group}_${key}, which has higher priority than the .ini configuration. Be careful not to use special characters in group name ([,], _).

Note

This function is included in nnstreamer internal header for native APIs. When changing the declaration, you should update the internal header (nnstreamer_internal.h).

Definition at line 609 of file nnstreamer_conf.c.

Here is the call graph for this function:



9.90.2.4 nnsconf_get_custom_value_string()

```
G_BEGIN_DECLS gchar* nnsconf_get_custom_value_string (
            const gchar * group,
            const gchar * key )
```

Get the custom configuration value from .ini and envvar. @detail For predefined configurations defined in this header, use the given enum for faster configuration processing. For custom configurations not defined in this header, you may use this API to access your own custom configurations. Configuration values may be loaded only once during runtime, thus, if the values are changed in run-time, the changes are not guaranteed to be reflected. The ENVVAR is supposed to be NNSTREAMER_${group}_${key}, which has higher priority than the .ini configuration. Be careful not to use special characters in group name ([,], _).

Parameters

in	*group*	The group name, [group], in .ini file.
in	*key*	The key name, key = value, in .ini file.

Returns

The newly allocated string. A caller must free it. NULL if it's not available.

Get the custom configuration value from .ini and envvar. @detail For predefined configurations defined in this header, use the given enum for faster configuration processing. For custom configurations not defined in this header, you may use this API to access your own custom configurations. Configuration values may be loaded only once during runtime, thus, if the values are changed in run-time, the changes are not guaranteed to be reflected. The ENVVAR is supposed to be NNSTREAMER_${group}_${key}, which has higher priority than the .ini configuration. Be careful not to use special characters in group name ([,], _).

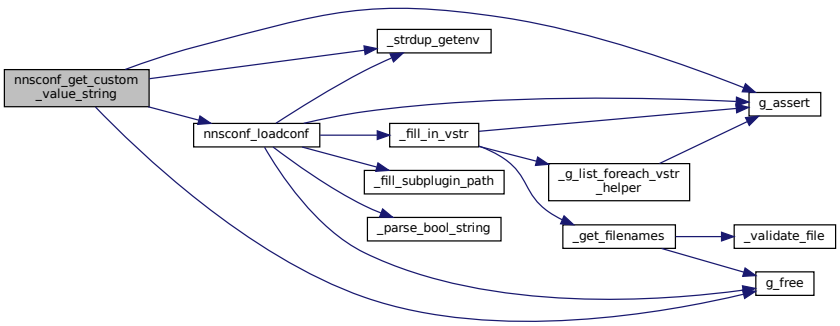
Note

This function is included in nnstreamer internal header for native APIs. When changing the declaration, you should update the internal header (nnstreamer_internal.h).

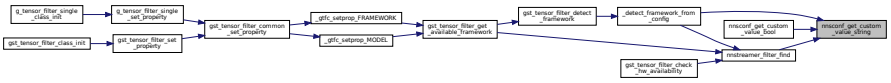
Internal lib error? out-of-memory?

Definition at line 557 of file nnstreamer_conf.c.

Here is the call graph for this function:



Here is the caller graph for this function:

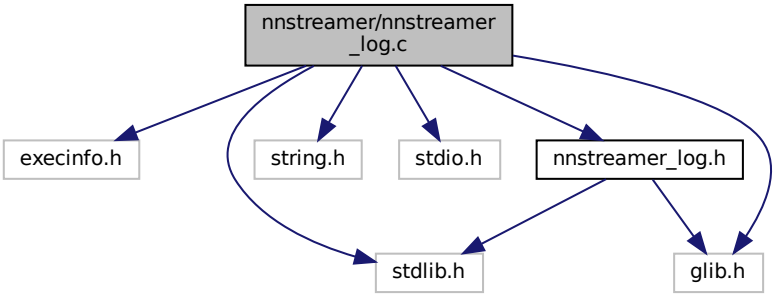


9.91 nnstreamer/nnstreamer_log.c File Reference

Internal log and error handling for NNStreamer plugins and core codes.

```
#include <execinfo.h>
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include <glib.h>
#include "nnstreamer_log.h"
```

Include dependency graph for nnstreamer_log.c:



Macros

- #define _NNSTREAMER_ERROR_LENGTH (4096U)

Functions

- char * _backtrace_to_string (void)

  stack trace as a string for error messages
- G_LOCK_DEFINE_STATIC (errlock)
- const char * _nnstreamer_error (void)

  return the last internal error string and clean it.
- __attribute__ ((__format__(__printf__, 1, 2)))

  overwrites the error message buffer with the new message.
- void _nnstreamer_error_clean (void)

  cleans up the error message buffer.

Variables

- static char errmsg [_NNSTREAMER_ERROR_LENGTH] = { 0 }
- static int errmsg_reported = 0

9.91.1 Detailed Description

Internal log and error handling for NNStreamer plugins and core codes.

Date

31 Mar 2022

See also

http://github.com/nnstreamer/nnstreamer

Author

MyungJoo Ham myungjoo.ham@samsung.com

Bug No known bugs except for NYI items

9.91.2 Macro Definition Documentation

9.91.2.1 _NNSTREAMER_ERROR_LENGTH

```
#define _NNSTREAMER_ERROR_LENGTH (4096U)
```

Definition at line 70 of file nnstreamer_log.c.

9.91.3 Function Documentation

9.91.3.1 __attribute__()

```
__attribute__ (
            (__format__(__printf__, 1, 2))  )
```

overwrites the error message buffer with the new message.

The attribute is for clang workaround in macos: https://stackoverflow.com/questions/20167124/vsprintf-a

Definition at line 97 of file nnstreamer_log.c.

9.91.3.2 _backtrace_to_string()

```
char* _backtrace_to_string (
            void  )
```

stack trace as a string for error messages

Returns

a string of stacktrace result. caller should free it.

Todo The .c file location of this function might be not appropriate.

Definition at line 35 of file nnstreamer_log.c.

9.91.3.3 _nnstreamer_error()

```
const char* _nnstreamer_error (
            void  )
```

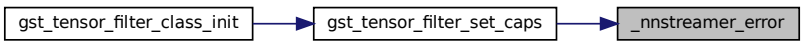
return the last internal error string and clean it.

Returns

a string of error. Do not free the returned string.

Definition at line 81 of file nnstreamer_log.c.

Here is the caller graph for this function:



9.91.3.4 _nnstreamer_error_clean()

```
void _nnstreamer_error_clean (
            void  )
```

cleans up the error message buffer.

Definition at line 120 of file nnstreamer_log.c.

9.91.3.5 G_LOCK_DEFINE_STATIC()

```
G_LOCK_DEFINE_STATIC (
             errlock  )
```

9.91.4 Variable Documentation

9.91.4.1 errmsg

```
char errmsg[_NNSTREAMER_ERROR_LENGTH] = { 0 }  [static]
```

Definition at line 71 of file nnstreamer_log.c.

9.91.4.2 errmsg_reported

```
int errmsg_reported = 0  [static]
```

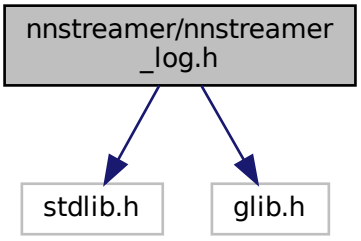
Definition at line 73 of file nnstreamer_log.c.

9.92 nnstreamer/nnstreamer_log.h File Reference

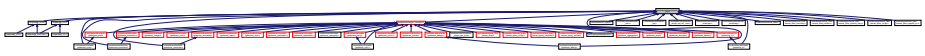
Internal log util for NNStreamer plugins and native APIs.

```
#include <stdlib.h>
#include <glib.h>
```

Include dependency graph for nnstreamer_log.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define TAG_NAME "nnstreamer"
- #define ml_logi g_info
- #define ml_logw g_warning
- #define ml_loge g_critical
- #define ml_logd g_debug
- #define ml_logf g_error
- #define GST_ELEMENT_ERROR_BTRACE(s, errtype, errcode, mesg)
- #define ml_logf_stacktrace(...)
- #define ml_log_stacktrace(logfunc, ...)
- #define ml_loge_stacktrace(...) ml_log_stacktrace(ml_loge, __VA_ARGS__)
- #define nns_logi ml_logi
- #define nns_logw ml_logw
- #define nns_loge ml_loge
- #define nns_logd ml_logd
- #define nns_logf ml_logf

Functions

- char * _backtrace_to_string (void)

  stack trace as a string for error messages
- const char * _nnstreamer_error (void)

  return the last internal error string and clean it.
- void _nnstreamer_error_write (const char *fmt,...)

  overwrites the error message buffer with the new message.
- void _nnstreamer_error_clean (void)

  cleans up the error message buffer.

9.92.1 Detailed Description

Internal log util for NNStreamer plugins and native APIs.

Date

12 Mar 2020

See also

http://github.com/nnstreamer/nnstreamer

Author

Jaeyun Jung jy1210.jung@samsung.com

Bug No known bugs except for NYI items

9.92.2 Macro Definition Documentation

9.92.2.1 GST_ELEMENT_ERROR_BTRACE

```
#define GST_ELEMENT_ERROR_BTRACE(
              s,
              errtype,
              errcode,
              mesg )
```

Value:

```
      do { \
      char *btrace = _backtrace_to_string (); \
      if (btrace) { \
        GST_ELEMENT_ERROR (s, errtype, errcode, mesg, ("%s", btrace)); \
        free (btrace); \
      } \
    } while (0)
```

Definition at line 94 of file nnstreamer_log.h.

9.92.2.2 ml_log_stacktrace

```
#define ml_log_stacktrace(
              logfunc,
              ... )
```

Value:

```
      do { \
      char *btrace = _backtrace_to_string (); \
      if (btrace) { \
        logfunc ("%s\n", btrace);  \
        free (btrace); \
      } \
      logfunc (__VA_ARGS__); \
    } while (0)
```

Definition at line 111 of file nnstreamer_log.h.

9.92.2.3 ml_logd

```
#define ml_logd g_debug
```

Definition at line 79 of file nnstreamer_log.h.

9.92.2.4 ml_loge

```
#define ml_loge g_critical
```

Definition at line 78 of file nnstreamer_log.h.

9.92.2.5 ml_loge_stacktrace

```
#define ml_loge_stacktrace(
               ...  ) ml_log_stacktrace(ml_loge, __VA_ARGS__)
```

Definition at line 119 of file nnstreamer_log.h.

9.92.2.6 ml_logf

```
#define ml_logf g_error
```

Definition at line 80 of file nnstreamer_log.h.

9.92.2.7 ml_logf_stacktrace

```
#define ml_logf_stacktrace(
               ...  )
```

Value:

```
      do { \
      char *btrace = _backtrace_to_string (); \
      if (btrace) { \
        ml_loge ("%s\n", btrace);  \
        free (btrace); \
      } \
      ml_logf (__VA_ARGS__); \
    } while (0)
```

Definition at line 102 of file nnstreamer_log.h.

9.92.2.8 ml_logi

```
#define ml_logi g_info
```

Definition at line 76 of file nnstreamer_log.h.

9.92.2.9 ml_logw

```
#define ml_logw g_warning
```

Definition at line 77 of file nnstreamer_log.h.

9.92.2.10 nns_logd

```
#define nns_logd ml_logd
```

Definition at line 143 of file nnstreamer_log.h.

9.92.2.11 nns_loge

```
#define nns_loge ml_loge
```

Definition at line 142 of file nnstreamer_log.h.

9.92.2.12 nns_logf

```
#define nns_logf ml_logf
```

Definition at line 144 of file nnstreamer_log.h.

9.92.2.13 nns_logi

```
#define nns_logi ml_logi
```

Definition at line 140 of file nnstreamer_log.h.

9.92.2.14 nns_logw

```
#define nns_logw ml_logw
```

Definition at line 141 of file nnstreamer_log.h.

9.92.2.15 TAG_NAME

```
#define TAG_NAME "nnstreamer"
```

Definition at line 28 of file nnstreamer_log.h.

9.92.3 Function Documentation

9.92.3.1 _backtrace_to_string()

```
char* _backtrace_to_string (
            void  )
```

stack trace as a string for error messages

Returns

a string of stacktrace result. caller should free it.
a string of stacktrace result. caller should free it.

Todo The .c file location of this function might be not appropriate.

Definition at line 35 of file nnstreamer_log.c.

9.92.3.2 _nnstreamer_error()

```
const char* _nnstreamer_error (
            void  )
```

return the last internal error string and clean it.

Returns

a string of error. Do not free the returned string.

Definition at line 81 of file nnstreamer_log.c.

Here is the caller graph for this function:

9.92.3.3 _nnstreamer_error_clean()

```
void _nnstreamer_error_clean (
            void  )
```

cleans up the error message buffer.

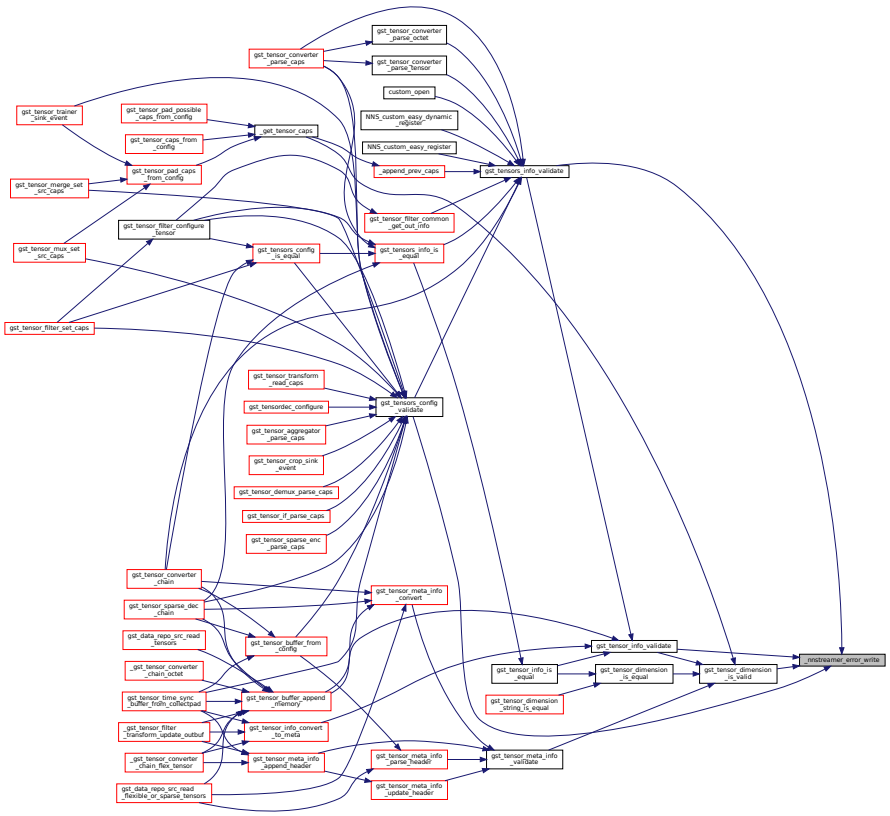
Definition at line 120 of file nnstreamer_log.c.

9.92.3.4 _nnstreamer_error_write()

```
void _nnstreamer_error_write (
            const char * fmt,
             ...  )
```

overwrites the error message buffer with the new message.

Here is the caller graph for this function:

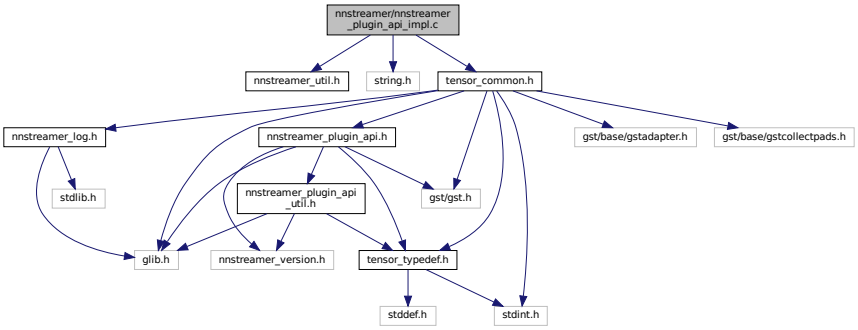


9.93 nnstreamer/nnstreamer_plugin_api_impl.c File Reference

Common data for NNStreamer, the GStreamer plugin for neural networks.

```
#include <nnstreamer_util.h>
#include <string.h>
#include <tensor_common.h>
```

Include dependency graph for nnstreamer_plugin_api_impl.c:



Classes

- struct GstTensorExtraInfo

  Data structure to describe a "extra" tensor data. This represents the information of the NNS_TENSOR_SIZE_LIMIT-th memory block for tensor stream.
- struct gst_tensor_aggregation_data_s

  Internal struct to handle aggregation data in hash table.

Macros

- #define NNS_TENSOR_RANK_LIMIT_PREV (4)

  The old rank of tensor.
- #define NNS_TENSOR_EXTRA_MAGIC 0xf00dc0de
- #define AGGREGATION_DEFAULT_KEY 0xC0FFEEU

Functions

- static gboolean gst_memory_map_is_extra_tensor (GstMapInfo *map)

  Check if given memory has extra tensors.
- static void gst_tensor_extra_info_init (GstTensorExtraInfo *extra, gsize reserved_size)

  Initialize GstTensorExtraInfo structure with given memory.
- tensor_time_sync_mode gst_tensor_time_sync_get_mode (const gchar *str)

  Get the corresponding mode from the string value.
- const gchar * gst_tensor_time_sync_get_mode_string (tensor_time_sync_mode mode)

  Get the time-sync mode string.
- gboolean gst_tensor_time_sync_set_option_data (tensor_time_sync_data *sync)

  Setup time sync option.

- static gboolean _gst_tensor_time_sync_is_eos (GstCollectPads *collect, tensor_time_sync_data *sync, guint empty)

  Internal function to detect EOS using the number of empty pads.
- gboolean gst_tensor_time_sync_get_current_time (GstCollectPads *collect, tensor_time_sync_data *sync, GstClockTime *current_time, GstBuffer *tensors_buf)

  A function call to decide current timestamp among collected pads based on PTS. It will decide current timestamp according to sync option. GstMeta is also copied with same sync mode.
- void gst_tensor_time_sync_flush (GstCollectPads *collect)

  A function to be called while processing a flushing event. It should clear old buffer and reset pad data.
- static gboolean _gst_tensor_time_sync_buffer_update (GstCollectPads *collect, GstCollectData *data, GstClockTime current, GstClockTime base, tensor_time_sync_data *sync)

  Internal function to update buffer in pad data based on the sync mode.
- gboolean gst_tensor_time_sync_buffer_from_collectpad (GstCollectPads *collect, tensor_time_sync_data *sync, GstClockTime current_time, GstBuffer *tensors_buf, GstTensorsConfig *configs, gboolean *is_eos)

  A function call to make tensors from collected pads. It decide which buffer is going to be used according to sync option.
- GstBuffer * gst_tensor_buffer_from_config (GstBuffer *in, GstTensorsConfig *config)

  Configure gst-buffer with tensors information. NNStreamer handles single memory chunk as single tensor. If incoming buffer has invalid memories, separate it and generate new gst-buffer using tensors information. Note that this function always takes the ownership of input buffer.
- static void gst_tensor_aggregation_free_data (gpointer data)

  Internal function to free aggregation data.
- static gst_tensor_aggregation_data_s * gst_tensor_aggregation_add_data (GHashTable *table, const guint32 key)

  Internal function to add new aggregation data.
- static gst_tensor_aggregation_data_s * gst_tensor_aggregation_get_data (GHashTable *table, const guint32 key)

  Internal function to get aggregation data.
- static void gst_tensor_aggregation_clear_internal (gpointer key, gpointer value, gpointer user_data)

  Internal function to remove all buffers from aggregation data.
- GHashTable * gst_tensor_aggregation_init (void)

  Gets new hash table for tensor aggregation.
- void gst_tensor_aggregation_clear (GHashTable *table, const guint32 key)

  Clears buffers from adapter.
- void gst_tensor_aggregation_clear_all (GHashTable *table)

  Clears buffers from all adapters in hash table.
- GstAdapter * gst_tensor_aggregation_get_adapter (GHashTable *table, const guint32 key)

  Gets adapter from hash table.
- static gboolean _append_prev_caps (const GstTensorsConfig *config)

  Internal function to check tensor dimensions to append old caps for backward compatibility (rank 4).
- static GstCaps * _get_tensor_caps (const GstTensorsConfig *config)

  Internal function to get caps for single tensor from config.
- static GstCaps * _get_tensors_caps (const GstTensorsConfig *config)

  Internal function to get caps for multi tensors from config.
- static GstCaps * _get_flexible_caps (const GstTensorsConfig *config)

  Internal function to get caps for flexible tensor from config.
- gboolean gst_structure_is_tensor_stream (const GstStructure *structure)

  Check given mimetype is tensor stream.
- media_type gst_structure_get_media_type (const GstStructure *structure)

  Get media type from structure.
- gboolean gst_tensors_config_from_peer (GstPad *pad, GstTensorsConfig *config, gboolean *is_fixed)

  Parse caps from peer pad and set tensors config.

- static gboolean _is_structure_dimension_same (GstStructure *st1, GstStructure *st2, const gchar *fieldname)

  Check whether two structures have the same dimension.
- void gst_tensor_caps_update_dimension (GstCaps *caps, GstCaps *filter)

  Update caps dimensions for negotiation.
- gboolean gst_tensor_caps_can_intersect (GstCaps *caps1, GstCaps *caps2)

  Try intersecting @caps1 and @caps2 for tensor stream.
- GstCaps * gst_tensor_pad_caps_from_config (GstPad *pad, const GstTensorsConfig *config)

  Get pad caps from tensors config and caps of the peer connected to the pad.
- GstCaps * gst_tensor_pad_possible_caps_from_config (GstPad *pad, const GstTensorsConfig *config)

  Get all possible caps from tensors config. Unlike gst_tensor_pad_caps_from_config(), this function does not check peer caps.
- tensor_format gst_tensor_pad_get_format (GstPad *pad)

  Get tensor format of current pad caps.
- GstCaps * gst_tensors_caps_from_config (const GstTensorsConfig *config)

  Get caps from tensors config (for other/tensors)
- GstCaps * gst_tensor_caps_from_config (const GstTensorsConfig *config)

  Get tensor caps from tensors config.
- gboolean gst_tensors_config_from_structure (GstTensorsConfig *config, const GstStructure *structure)

  Parse structure and set tensors config (for other/tensors)
- gboolean gst_tensor_meta_info_parse_memory (GstTensorMetaInfo *meta, GstMemory *mem)

  Parse memory and fill the tensor meta.
- GstMemory * gst_tensor_meta_info_append_header (GstTensorMetaInfo *meta, GstMemory *mem)

  Append header to memory.
- GstMemory * gst_tensor_buffer_get_nth_memory (GstBuffer *buffer, const guint index)

  Get the nth GstMemory from given buffer.
- gboolean gst_tensor_buffer_append_memory (GstBuffer *buffer, GstMemory *memory, const GstTensorInfo *info)

  Append memory to given buffer.
- guint gst_tensor_buffer_get_count (GstBuffer *buffer)

  Get the number of tensors in the buffer.
- static void set_property_value (GValue *prop_value, const GParamSpec *param_spec, const gchar *property_value)

  Sets the value of a property based on the specified property value and GParamSpec.
- void gst_tensor_parse_config_file (const gchar *config_path, const GObject *object)

  Parses a configuration file and sets the corresponding properties on a GObject.

Variables

- static const gchar * gst_tensor_time_sync_mode_string [ ]

9.93.1 Detailed Description

Common data for NNStreamer, the GStreamer plugin for neural networks.

NNStreamer Common Header's Contents (pipeline extension) Copyright (C) 2020 MyungJoo Ham myungjoo.↩ham@samsung.com

Date

14 Apr 2020

See also

https://github.com/nnstreamer/nnstreamer

Author

MyungJoo Ham myungjoo.ham@samsung.com

Bug No known bugs except for NYI items

9.93.2 Macro Definition Documentation

9.93.2.1 AGGREGATION_DEFAULT_KEY

```
#define AGGREGATION_DEFAULT_KEY 0xC0FFEEU
```

Definition at line 662 of file nnstreamer_plugin_api_impl.c.

9.93.2.2 NNS_TENSOR_EXTRA_MAGIC

```
#define NNS_TENSOR_EXTRA_MAGIC 0xf00dc0de
```

Definition at line 33 of file nnstreamer_plugin_api_impl.c.

9.93.2.3 NNS_TENSOR_RANK_LIMIT_PREV

```
#define NNS_TENSOR_RANK_LIMIT_PREV (4)
```

The old rank of tensor.

Definition at line 31 of file nnstreamer_plugin_api_impl.c.

9.93.3 Function Documentation

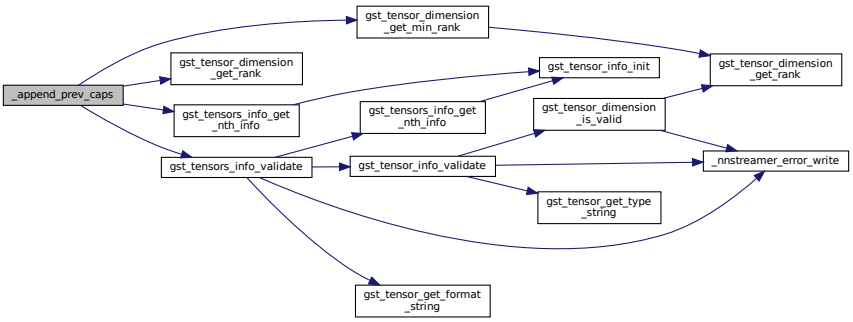
9.93.3.1 _append_prev_caps()

```
static gboolean _append_prev_caps (
            const GstTensorsConfig * config )  [static]
```

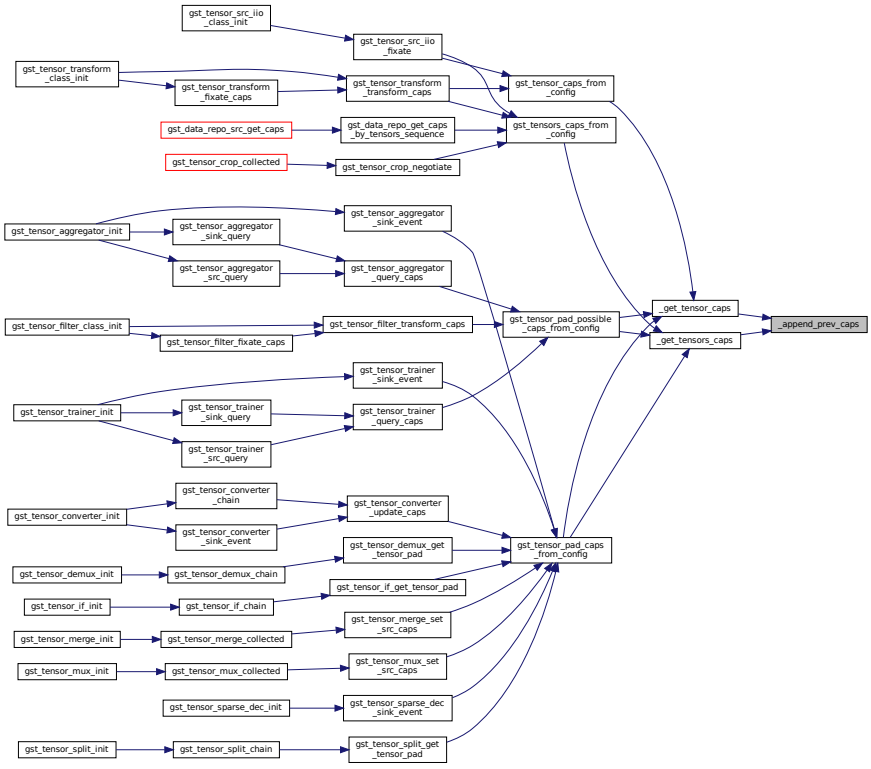
Internal function to check tensor dimensions to append old caps for backward compatibility (rank 4).

Definition at line 809 of file nnstreamer_plugin_api_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



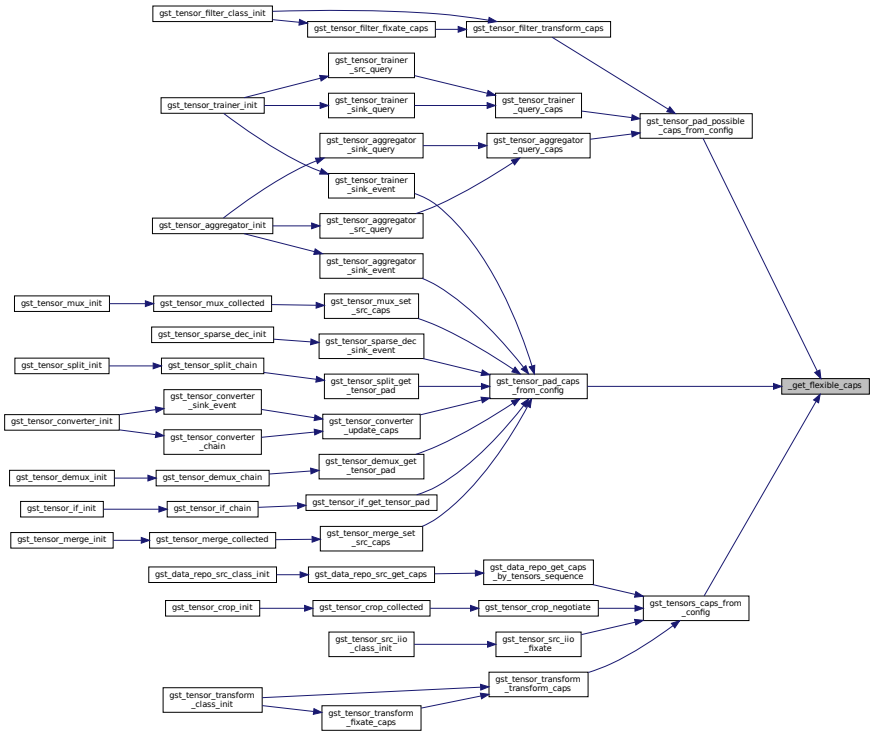
9.93.3.2 _get_flexible_caps()

```
static GstCaps* _get_flexible_caps (
            const GstTensorsConfig * config )  [static]
```

Internal function to get caps for flexible tensor from config.

Definition at line 964 of file nnstreamer_plugin_api_impl.c.

Here is the caller graph for this function:



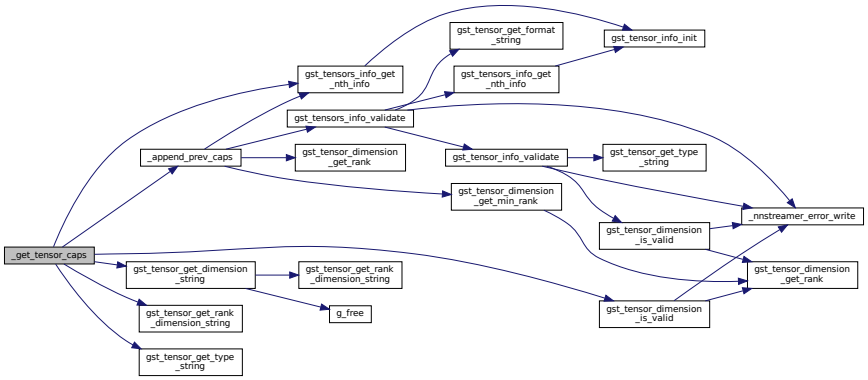
9.93.3.3 _get_tensor_caps()

```
static GstCaps* _get_tensor_caps (
            const GstTensorsConfig * config )  [static]
```

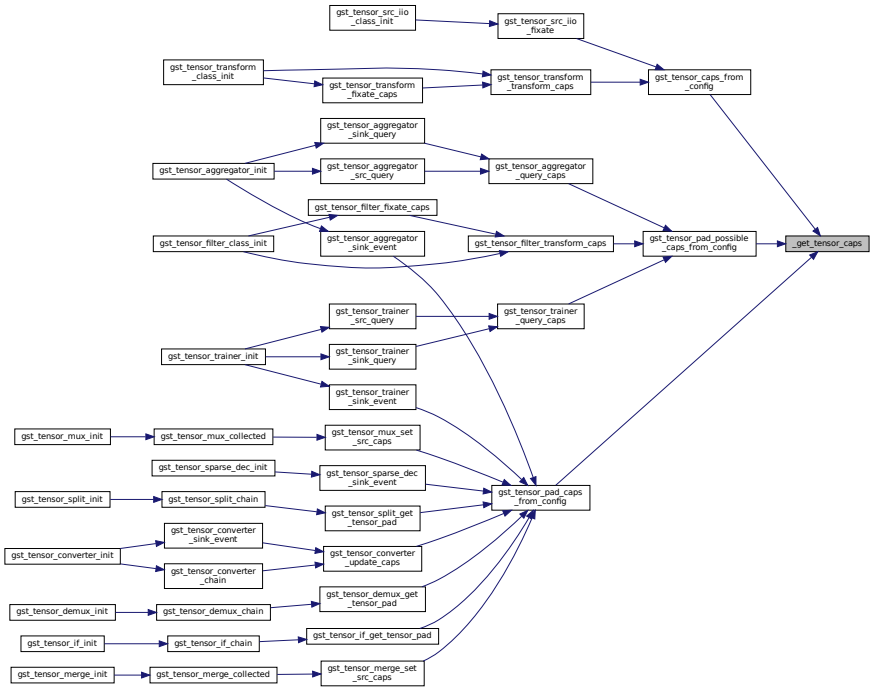
Internal function to get caps for single tensor from config.

Definition at line 839 of file nnstreamer_plugin_api_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



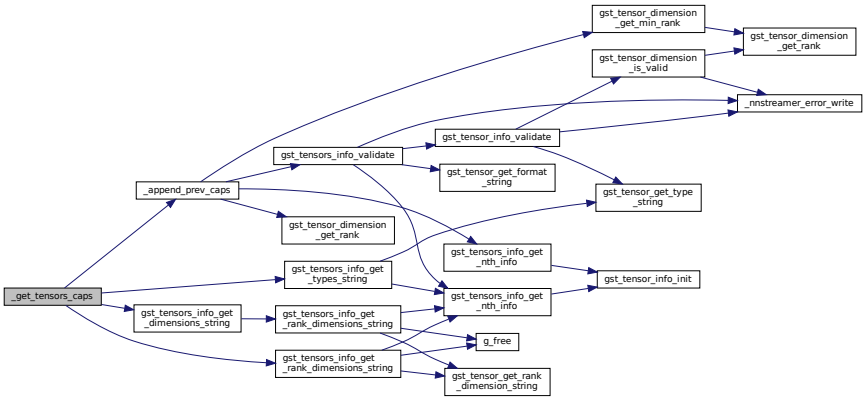
9.93.3.4 _get_tensors_caps()

```
static GstCaps* _get_tensors_caps (
            const GstTensorsConfig * config )  [static]
```

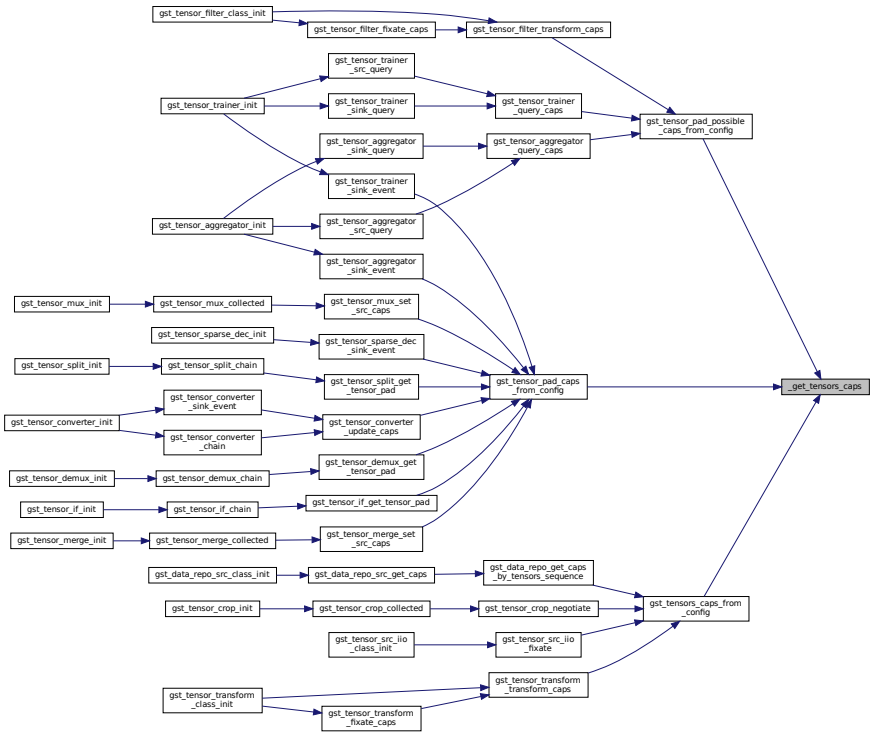
Internal function to get caps for multi tensors from config.

Definition at line 904 of file nnstreamer_plugin_api_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.93.3.5 _gst_tensor_time_sync_buffer_update()

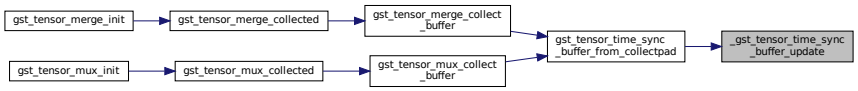
```
static gboolean _gst_tensor_time_sync_buffer_update (
            GstCollectPads * collect,
            GstCollectData * data,
```

```
            GstClockTime current,
            GstClockTime base,
            tensor_time_sync_data * sync )  [static]
```

Internal function to update buffer in pad data based on the sync mode.

Definition at line 287 of file nnstreamer_plugin_api_impl.c.

Here is the caller graph for this function:



9.93.3.6 _gst_tensor_time_sync_is_eos()

```
static gboolean _gst_tensor_time_sync_is_eos (
            GstCollectPads * collect,
            tensor_time_sync_data * sync,
            guint empty )  [static]
```

Internal function to detect EOS using the number of empty pads.

Parameters

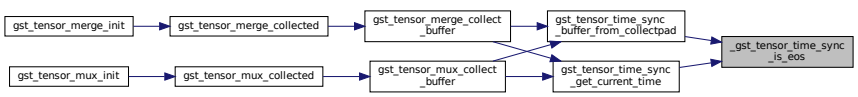
in	*collect*	Collect pad.
in	*sync*	Synchronization option.
in	*empty*	The number of empty pads (pad has no buffer).

Returns

True if EOS.

Definition at line 175 of file nnstreamer_plugin_api_impl.c.

Here is the caller graph for this function:



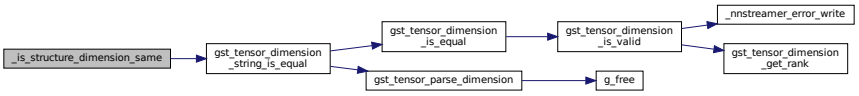
9.93.3.7 _is_structure_dimension_same()

```
static gboolean _is_structure_dimension_same (
            GstStructure * st1,
            GstStructure * st2,
            const gchar * fieldname )  [static]
```

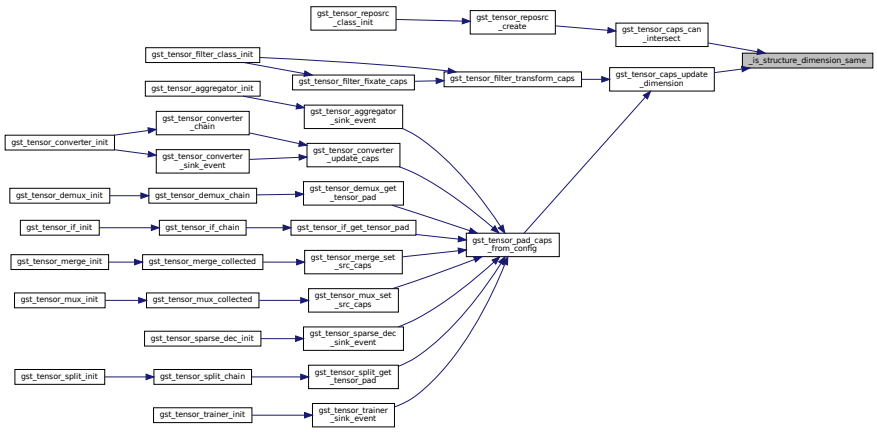
Check whether two structures have the same dimension.

Definition at line 1072 of file nnstreamer_plugin_api_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.93.3.8 gst_memory_map_is_extra_tensor()

```
static gboolean gst_memory_map_is_extra_tensor (
            GstMapInfo * map )  [static]
```

Check if given memory has extra tensors.

Parameters

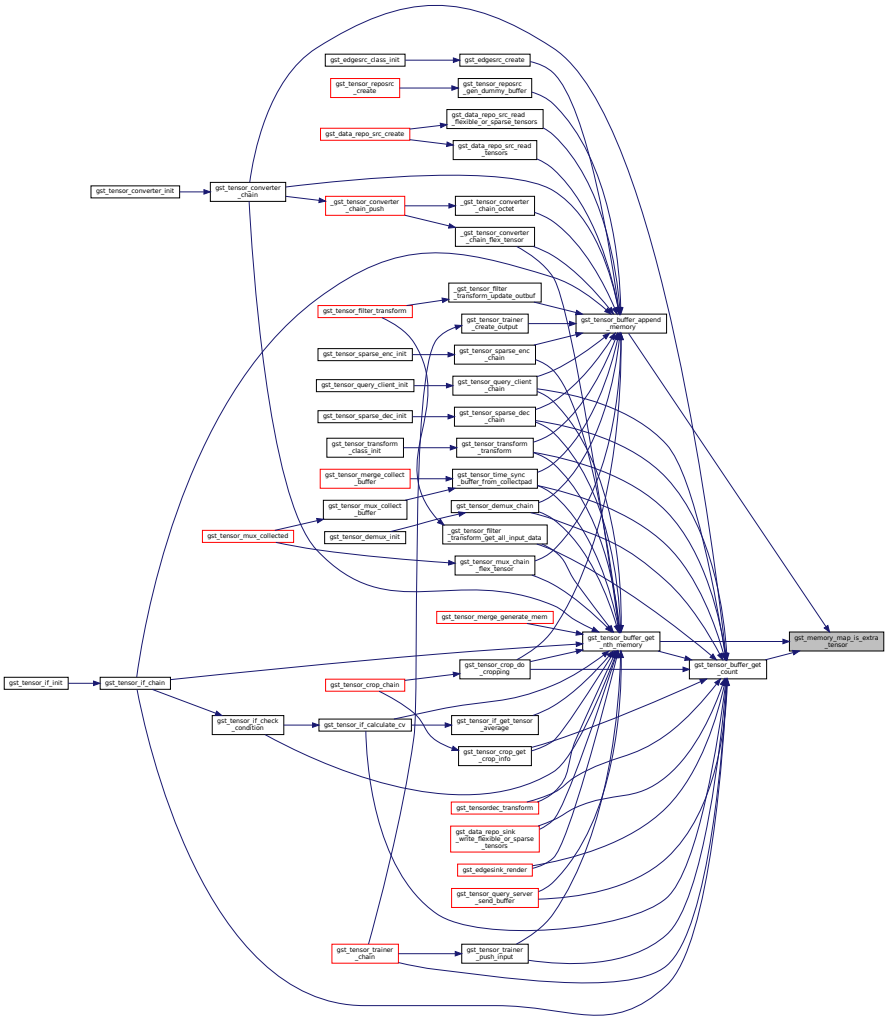
in	*map*	GstMapInfo of GstMemory to be checked.

Returns

TRUE if @map has extra tensors, otherwise FALSE.

Definition at line 54 of file nnstreamer_plugin_api_impl.c.

Here is the caller graph for this function:



9.93.3.9 gst_structure_get_media_type()

```
media_type gst_structure_get_media_type (
            const GstStructure * structure )
```

Get media type from structure.

Parameters

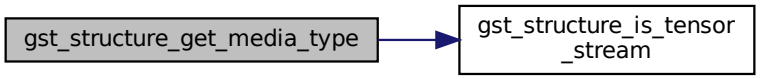
structure	structure to be interpreted

Returns

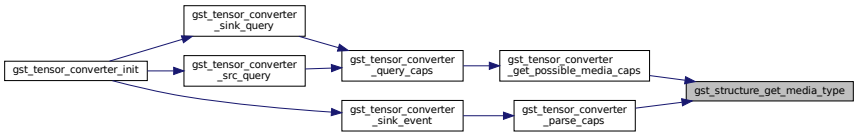
corresponding media type (returns _NNS_MEDIA_INVALID for unsupported type)

Definition at line 1001 of file nnstreamer_plugin_api_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.93.3.10 gst_structure_is_tensor_stream()

```
gboolean gst_structure_is_tensor_stream (
            const GstStructure * structure )
```

Check given mimetype is tensor stream.

Parameters

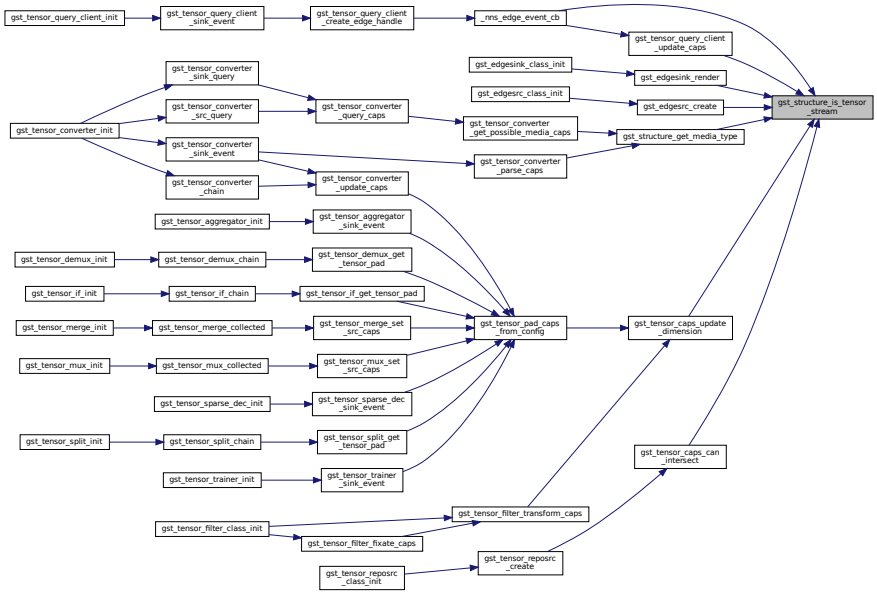
structure	structure to be interpreted

Returns

TRUE if mimetype is tensor stream

Definition at line 984 of file nnstreamer_plugin_api_impl.c.

Here is the caller graph for this function:



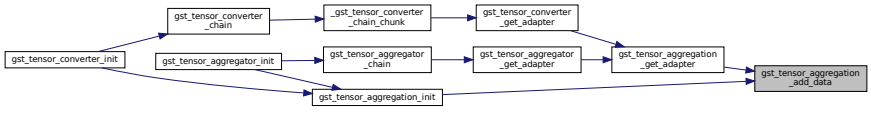
9.93.3.11 gst_tensor_aggregation_add_data()

```
static gst_tensor_aggregation_data_s* gst_tensor_aggregation_add_data (
            GHashTable * table,
            const guint32 key )  [static]
```

Internal function to add new aggregation data.

Definition at line 685 of file nnstreamer_plugin_api_impl.c.

Here is the caller graph for this function:



9.93.3.12 gst_tensor_aggregation_clear()

```
void gst_tensor_aggregation_clear (
            GHashTable * table,
            const guint32 key )
```

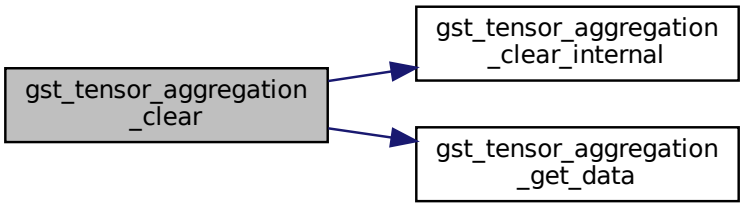
Clears buffers from adapter.

Parameters

table	a hash table instance initialized with gst_tensor_aggregation_init()
key	the key to look up (set null to get default adapter)

Definition at line 763 of file nnstreamer_plugin_api_impl.c.

Here is the call graph for this function:



9.93.3.13 gst_tensor_aggregation_clear_all()

```
void gst_tensor_aggregation_clear_all (
            GHashTable * table )
```

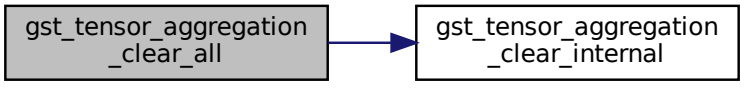
Clears buffers from all adapters in hash table.

Parameters

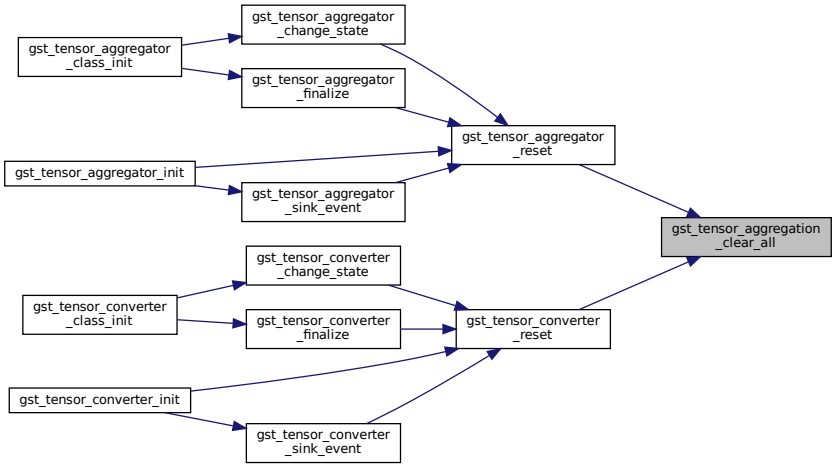
table	a hash table instance initialized with gst_tensor_aggregation_init()

Definition at line 778 of file nnstreamer_plugin_api_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



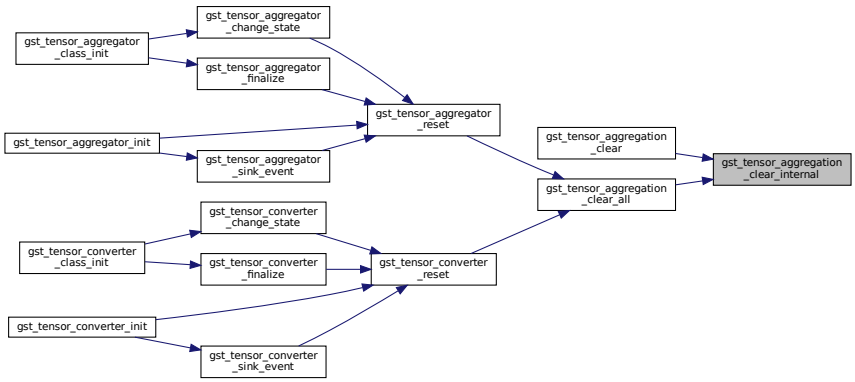
9.93.3.14 gst_tensor_aggregation_clear_internal()

```
static void gst_tensor_aggregation_clear_internal (
            gpointer key,
            gpointer value,
            gpointer user_data )  [static]
```

Internal function to remove all buffers from aggregation data.

Definition at line 718 of file nnstreamer_plugin_api_impl.c.

Here is the caller graph for this function:



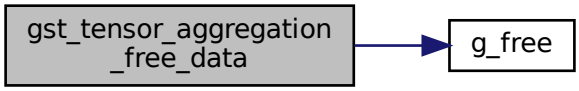
9.93.3.15 gst_tensor_aggregation_free_data()

```
static void gst_tensor_aggregation_free_data (
            gpointer data )  [static]
```

Internal function to free aggregation data.

Definition at line 668 of file nnstreamer_plugin_api_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:

9.93.3.16 gst_tensor_aggregation_get_adapter()

```
GstAdapter* gst_tensor_aggregation_get_adapter (
            GHashTable * table,
            const guint32 key )
```

Gets adapter from hash table.

Parameters

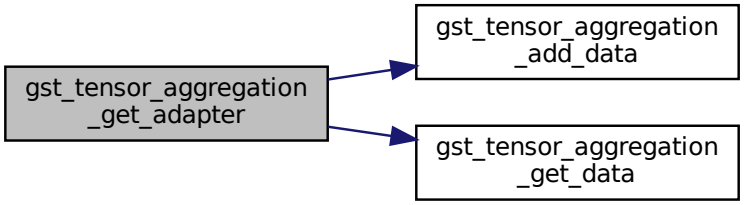
table	a hash table instance initialized with gst_tensor_aggregation_init()
key	the key to look up (set null to get default adapter)

Returns

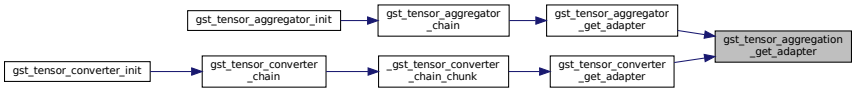
gst-adapter instance. DO NOT release this instance.

Definition at line 790 of file nnstreamer_plugin_api_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



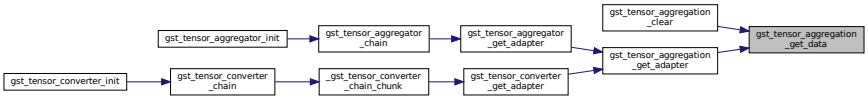
9.93.3.17 gst_tensor_aggregation_get_data()

```
static gst_tensor_aggregation_data_s* gst_tensor_aggregation_get_data (
            GHashTable * table,
            const guint32 key )  [static]
```

Internal function to get aggregation data.

Definition at line 706 of file nnstreamer_plugin_api_impl.c.

Here is the caller graph for this function:



9.93.3.18 gst_tensor_aggregation_init()

```
GHashTable* gst_tensor_aggregation_init (
            void  )
```

Gets new hash table for tensor aggregation.

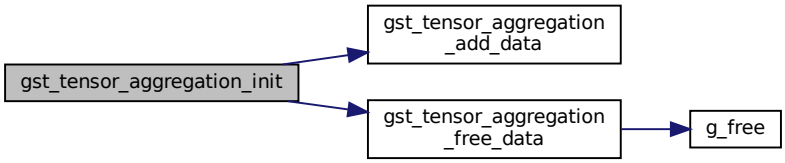
Returns

Newly allocated hash table, caller should release this using g_hash_table_destroy().

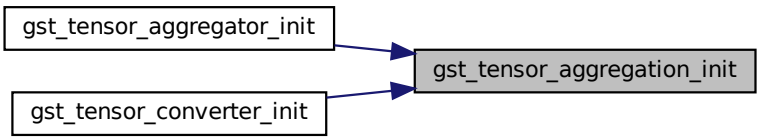
Add default adapter (for the case if buffer has no specific id). If gst-buffer has tensor-meta which includes client-id, e.g., aggregation frames from multiple clients on query-server pipeline, nnstreamer element should parse meta and request adapter with this id. However, on normal pipeline, gst-buffer does not contain tensor-meta, then the element may request adapter with null key string.

Definition at line 737 of file nnstreamer_plugin_api_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.93.3.19 gst_tensor_buffer_append_memory()

```
gboolean gst_tensor_buffer_append_memory (
            GstBuffer * buffer,
            GstMemory * memory,
            const GstTensorInfo * info )
```

Append *memory* to given *buffer*.

Parameters

	[in/out]	buffer GstBuffer to be appended.
in	*memory*	GstMemory to append. This function takes ownership of this, even if it returns failure.
in	*info*	GstTensorInfo of given *memory*.

Returns

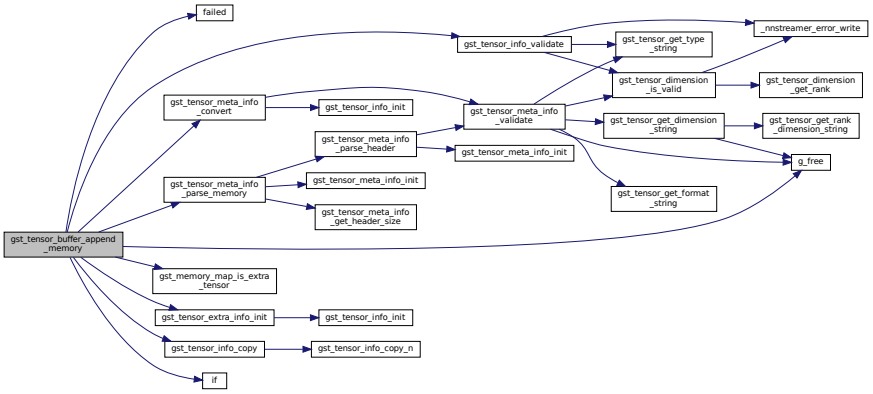
TRUE if successfully appended, otherwise FALSE.

Free the name string, cause it does not freed by gstreamer.

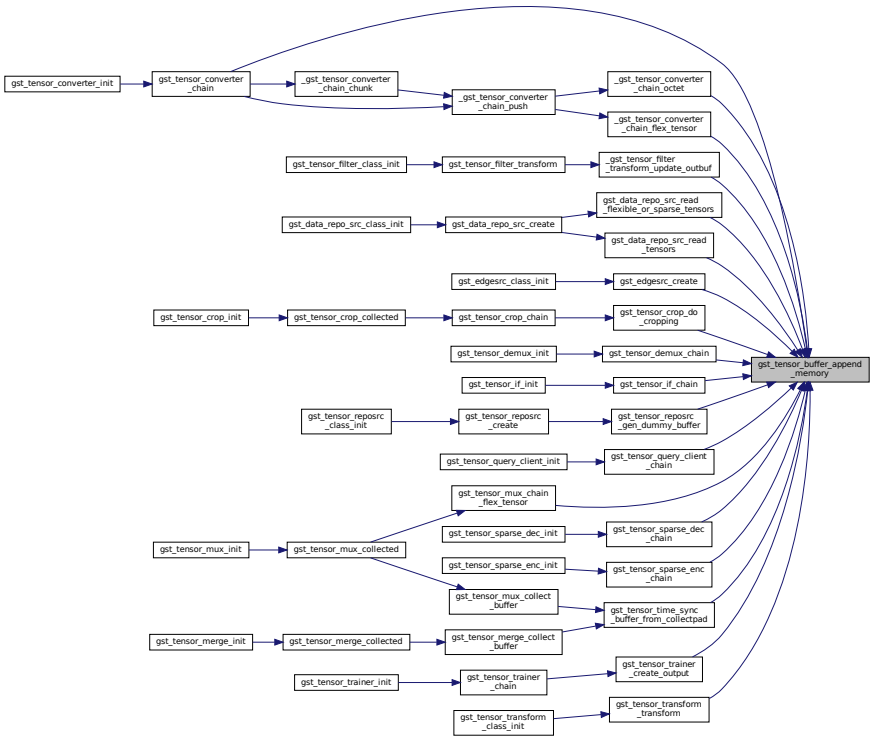
Todo Make custom gst_allocator later?

Definition at line 1666 of file nnstreamer_plugin_api_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.93.3.20 gst_tensor_buffer_from_config()

```
GstBuffer* gst_tensor_buffer_from_config (
            GstBuffer * in,
            GstTensorsConfig * config )
```

Configure gst-buffer with tensors information. NNStreamer handles single memory chunk as single tensor. If incoming buffer has invalid memories, separate it and generate new gst-buffer using tensors information. Note that this function always takes the ownership of input buffer.

Parameters

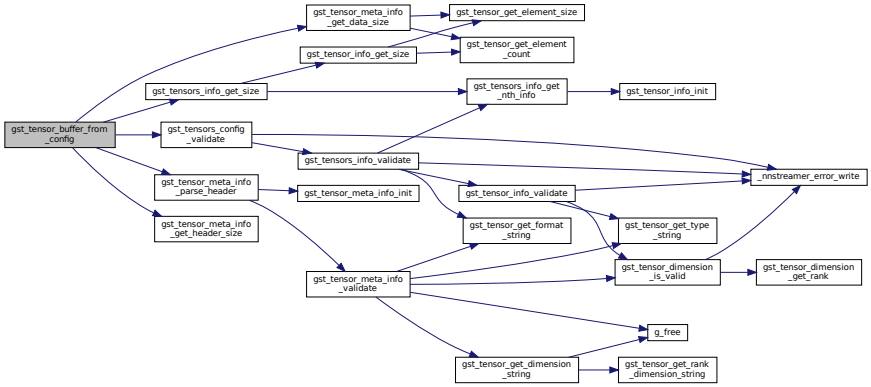
in	input buffer
config	tensors config structure

Returns

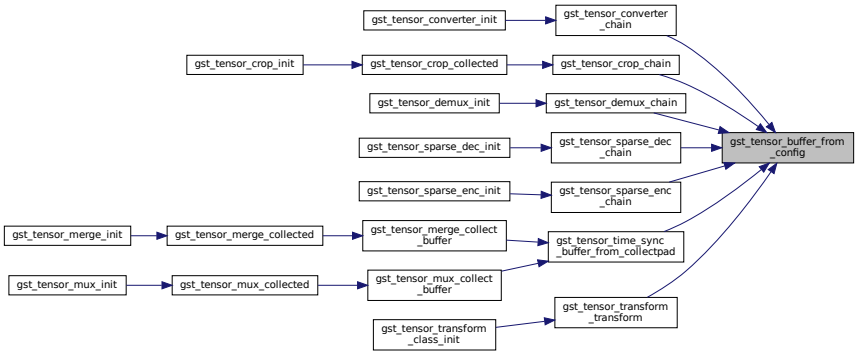
Newly allocated buffer. Null if failed. Caller should unref the buffer using gst_buffer_unref().

Definition at line 535 of file nnstreamer_plugin_api_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



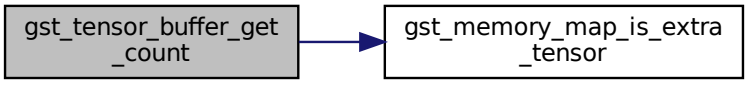
9.93.3.21 gst_tensor_buffer_get_count()

```
guint gst_tensor_buffer_get_count (
            GstBuffer * buffer )
```

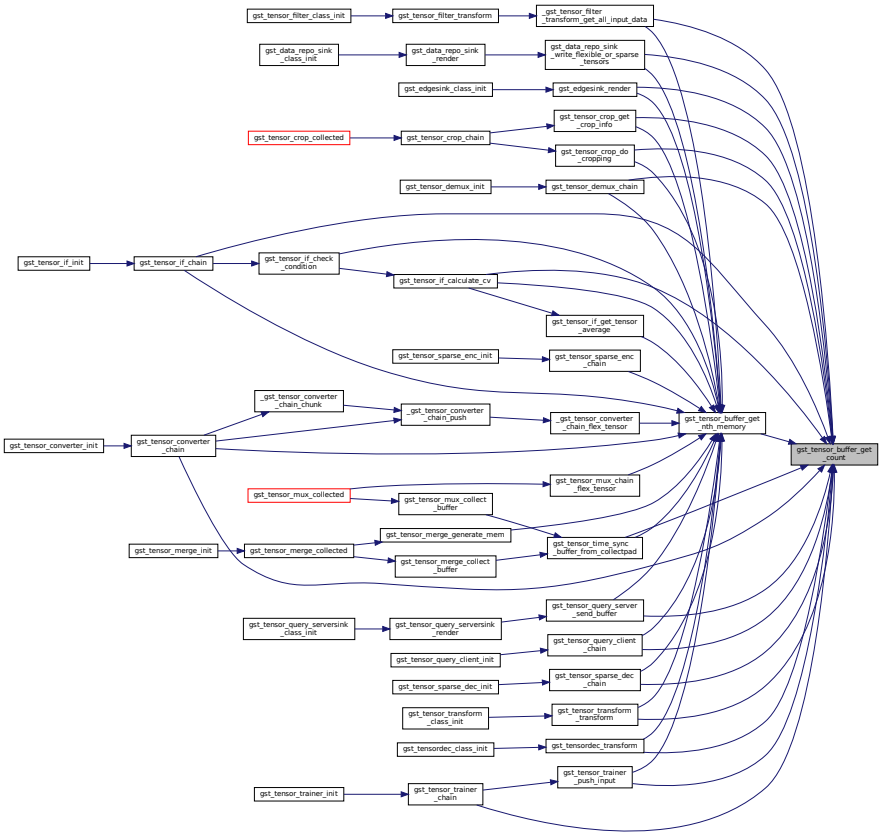
Get the number of tensors in the buffer.

Definition at line 1813 of file nnstreamer_plugin_api_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.93.3.22 gst_tensor_buffer_get_nth_memory()

```
GstMemory* gst_tensor_buffer_get_nth_memory (
            GstBuffer * buffer,
            const guint index )
```

Get the nth GstMemory from given *buffer*.

Parameters

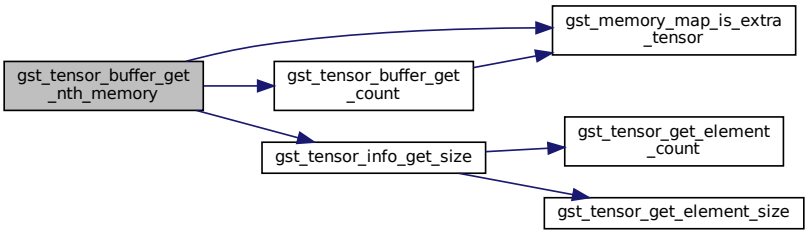
in	*buffer*	GstBuffer to be parsed.
in	*index*	Index of GstMemory to be returned.

Returns

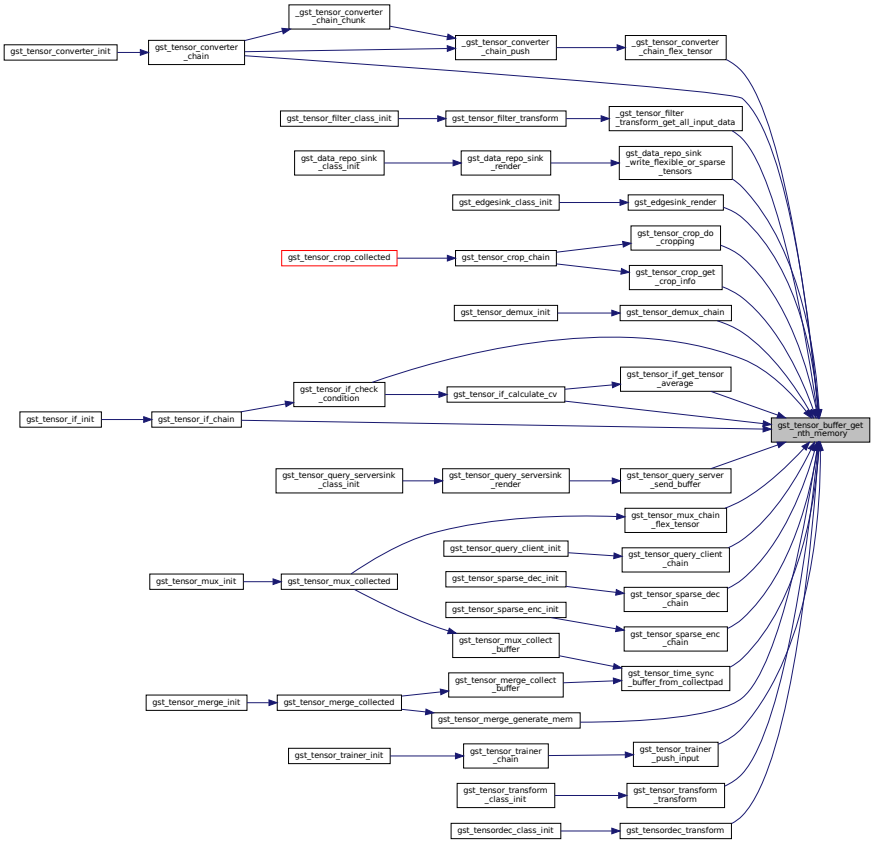
GstMemory if found, otherwise NULL (Caller should free returned memory using gst_memory_unref()).

Definition at line 1586 of file nnstreamer_plugin_api_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.93.3.23 gst_tensor_caps_can_intersect()

```
gboolean gst_tensor_caps_can_intersect (
            GstCaps * caps1,
            GstCaps * caps2 )
```

Try intersecting @caps1 and @caps2 for tensor stream.

Parameters

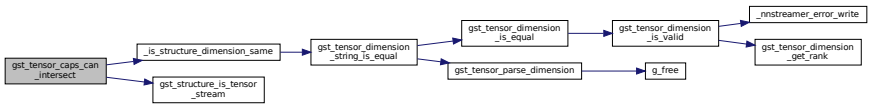
caps1	a GstCaps to intersect
caps2	a GstCaps to intersect

Returns

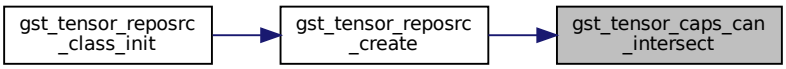
TRUE if intersection would be not empty.

Definition at line 1142 of file nnstreamer_plugin_api_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.93.3.24 gst_tensor_caps_from_config()

```
GstCaps* gst_tensor_caps_from_config (
            const GstTensorsConfig * config )
```

Get tensor caps from tensors config.

Get tensor caps from tensors config (for other/tensor)

Parameters

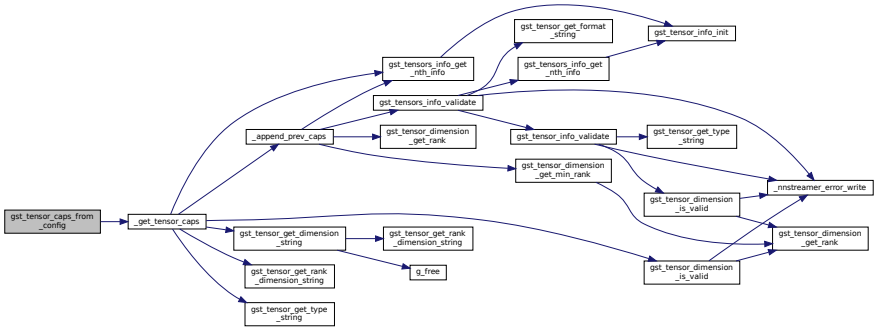
config	tensors config info

Returns

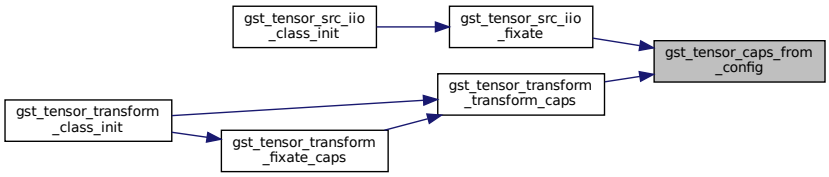
caps for given config

Definition at line 1395 of file nnstreamer_plugin_api_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.93.3.25 gst_tensor_caps_update_dimension()

```
void gst_tensor_caps_update_dimension (
            GstCaps * caps,
            GstCaps * filter )
```

Update caps dimensions for negotiation.

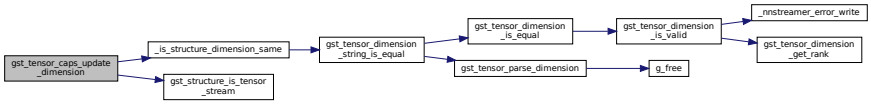
Update caps dimension for negotiation.

Parameters

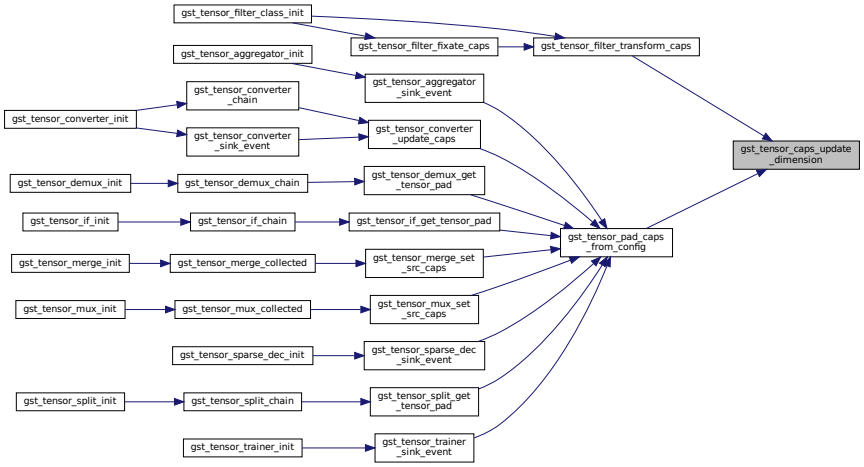
caps	caps to compare and update
filter	caps to compare

Definition at line 1093 of file nnstreamer_plugin_api_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.93.3.26 gst_tensor_extra_info_init()

```
static void gst_tensor_extra_info_init (
            GstTensorExtraInfo * extra,
            gsize reserved_size )  [static]
```

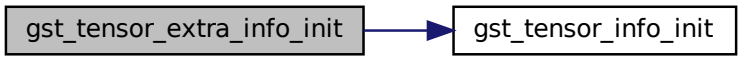
Initialize GstTensorExtraInfo structure with given *memory*.

Parameters

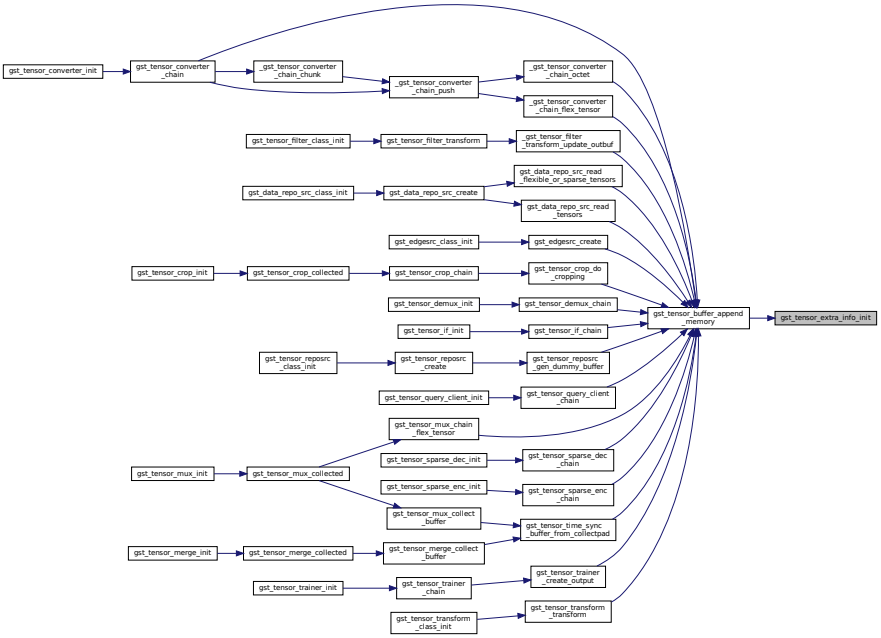
	[in/out]	extra GstTensorExtraInfo to be initialized.
in	*reserved_size*	The memory size of extra memory block.

Definition at line 78 of file nnstreamer_plugin_api_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.93.3.27 gst_tensor_meta_info_append_header()

```
GstMemory* gst_tensor_meta_info_append_header (
            GstTensorMetaInfo * meta,
            GstMemory * mem )
```

Append header to memory.

Parameters

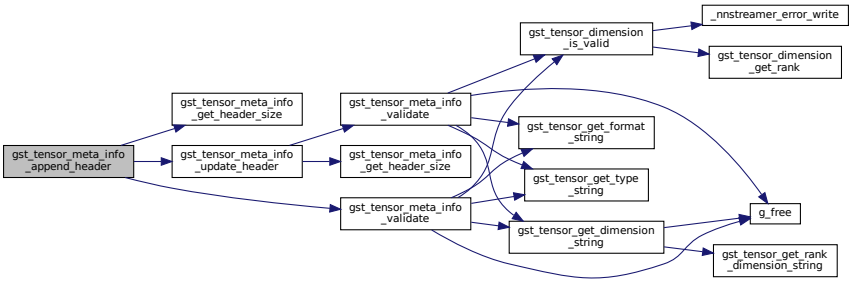
in	*meta*	tensor meta structure
in	*mem*	pointer to GstMemory

Returns

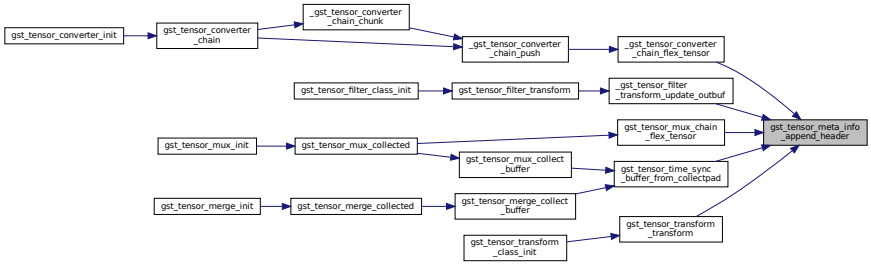
Newly allocated GstMemory (Caller should free returned memory using gst_memory_unref())

Definition at line 1544 of file nnstreamer_plugin_api_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.93.3.28 gst_tensor_meta_info_parse_memory()

```
gboolean gst_tensor_meta_info_parse_memory (
            GstTensorMetaInfo * meta,
            GstMemory * mem )
```

Parse memory and fill the tensor meta.

Parameters

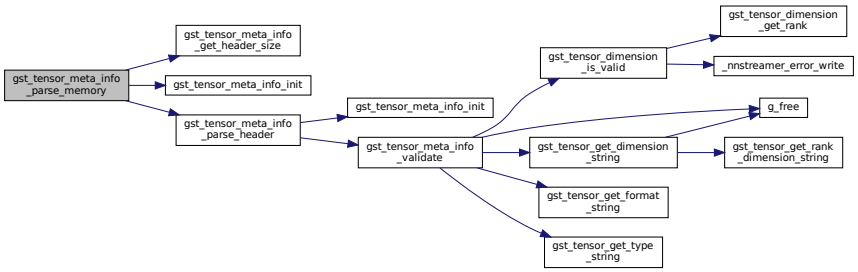
out	*meta*	tensor meta structure to be filled
in	*mem*	pointer to GstMemory to be parsed

Returns

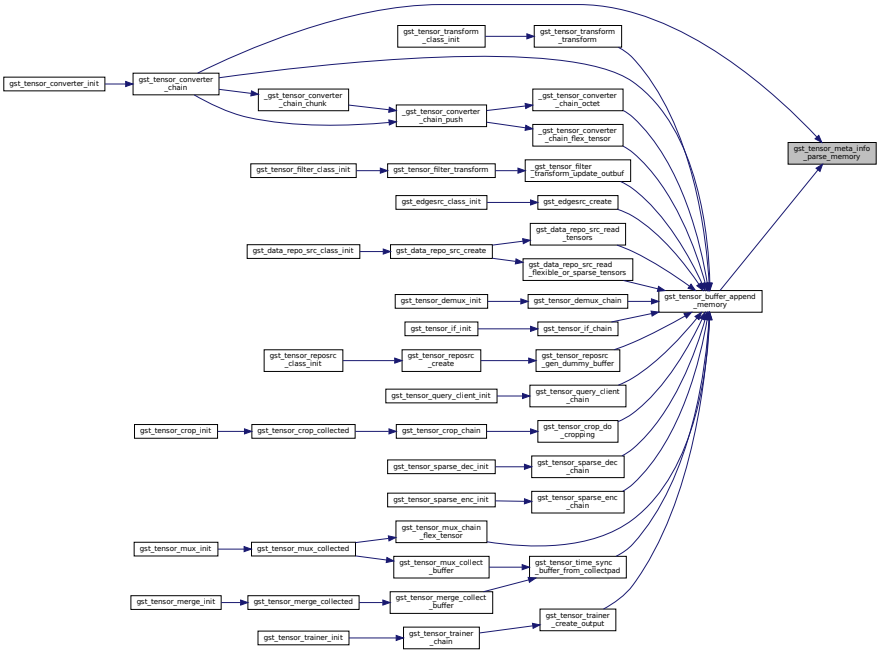
TRUE if successfully set the meta

Definition at line 1509 of file nnstreamer_plugin_api_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.93.3.29 gst_tensor_pad_caps_from_config()

```
GstCaps* gst_tensor_pad_caps_from_config (
            GstPad * pad,
            const GstTensorsConfig * config )
```

Get pad caps from tensors config and caps of the peer connected to the pad.

Parameters

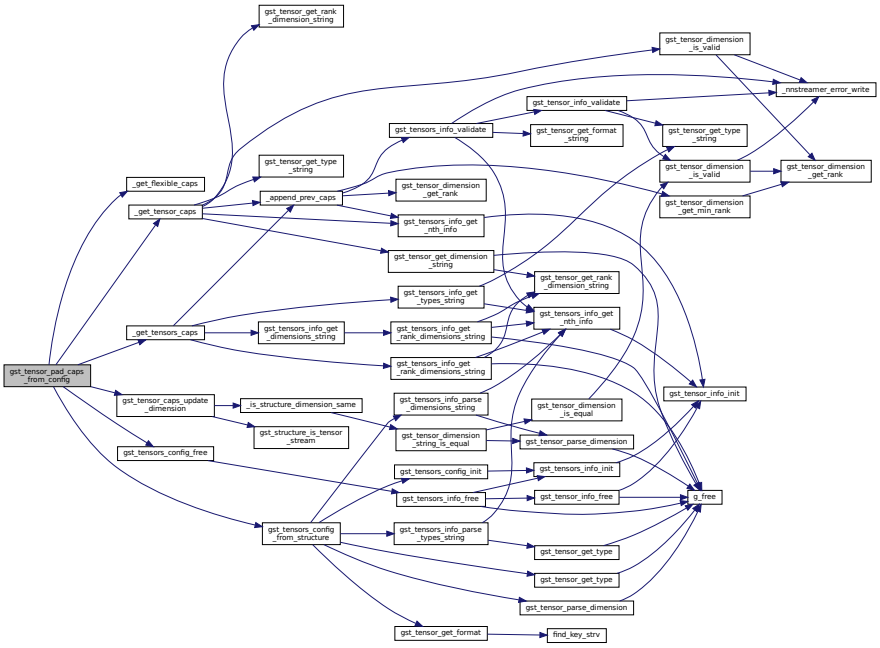
pad	GstPad to get possible caps
config	tensors config structure

Returns

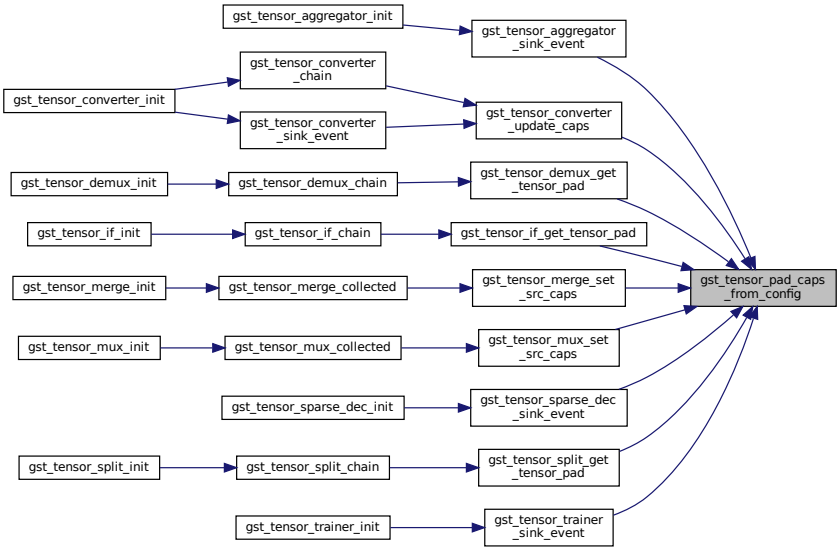
caps for given config. Caller is responsible for unreffing the returned caps.

Definition at line 1209 of file nnstreamer_plugin_api_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.93.3.30 gst_tensor_pad_get_format()

```
tensor_format gst_tensor_pad_get_format (
            GstPad * pad )
```

Get tensor format of current pad caps.

Parameters

pad	GstPad to check current caps.

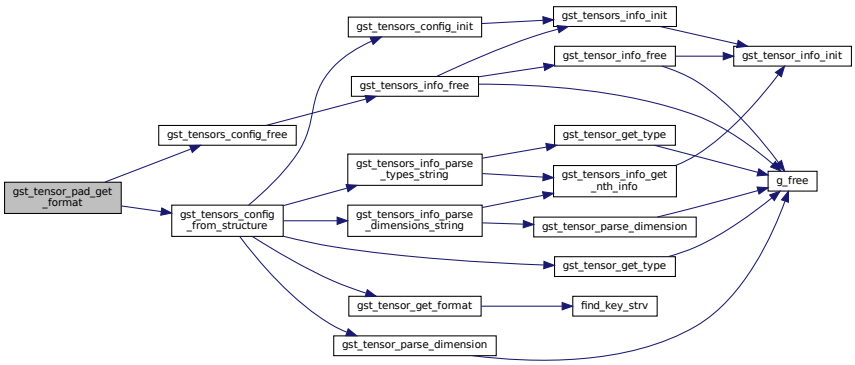
Returns

The tensor_format of current pad caps.

If pad does not have tensor caps return _NNS_TENSOR_FORMAT_END

Definition at line 1343 of file nnstreamer_plugin_api_impl.c.

Here is the call graph for this function:



9.93.3.31 gst_tensor_pad_possible_caps_from_config()

```
GstCaps* gst_tensor_pad_possible_caps_from_config (
            GstPad * pad,
            const GstTensorsConfig * config )
```

Get all possible caps from tensors config. Unlike gst_tensor_pad_caps_from_config(), this function does not check peer caps.

Parameters

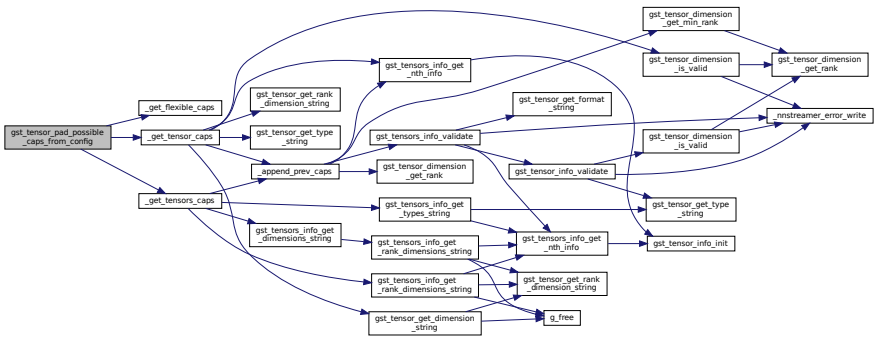
pad	GstPad to get possible caps
config	tensors config structure

Returns

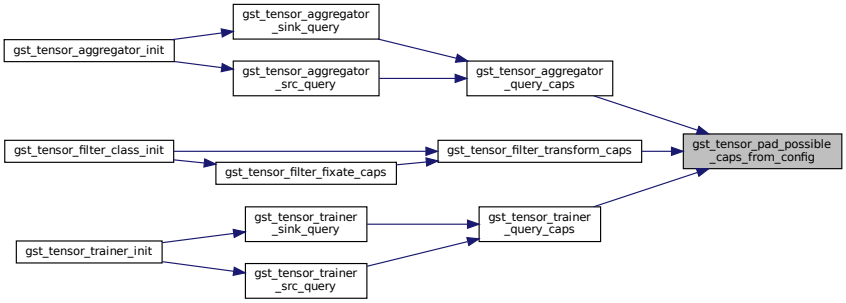
caps for given config. Caller is responsible for unreffing the returned caps.

Definition at line 1286 of file nnstreamer_plugin_api_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.93.3.32 gst_tensor_parse_config_file()

```
void gst_tensor_parse_config_file (
            const gchar * config_path,
            const GObject * object )
```

Parses a configuration file and sets the corresponding properties on a GObject.

This function reads the contents of the configuration file located at the given path and sets the properties of the specified GObject based on the configuration data.

Parameters

config_path	The path to the configuration file.
object	The GObject on which to set the properties.

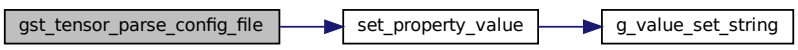
Note

The responsibility of managing the memory of the GObject passed as a parameter lies outside this function.

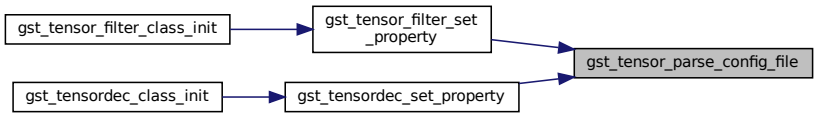
Iterate over each line

Definition at line 1902 of file nnstreamer_plugin_api_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.93.3.33 gst_tensor_time_sync_buffer_from_collectpad()

```
gboolean gst_tensor_time_sync_buffer_from_collectpad (
            GstCollectPads * collect,
            tensor_time_sync_data * sync,
            GstClockTime current_time,
            GstBuffer * tensors_buf,
            GstTensorsConfig * configs,
            gboolean * is_eos )
```

A function call to make tensors from collected pads. It decide which buffer is going to be used according to sync option.

A function call to make tensors from collected pads It decide which buffer is going to be used according to sync option.

Returns

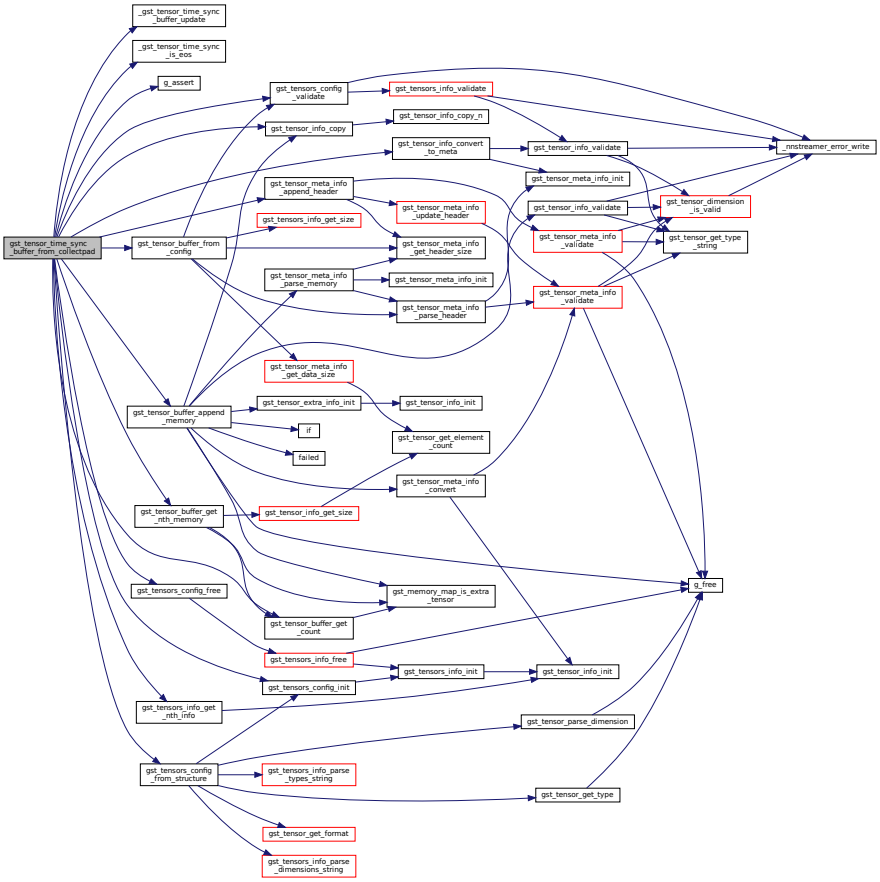
True to push buffer.

This would be an internal logic error. in_configs should be already confirmed valid at the negotiation phase and this function should be called in a running pipeline. If new sync mode is enabled (e.g., handle output when a pad gets new buffer), this may cause unexpected exception.

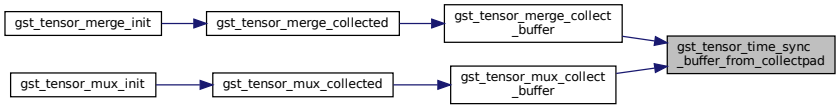
These are internal logic error. If given inputs are incorrect, the negotiation should have been failed before this stage.

Definition at line 332 of file nnstreamer_plugin_api_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.93.3.34 gst_tensor_time_sync_flush()

```
void gst_tensor_time_sync_flush (
            GstCollectPads * collect )
```

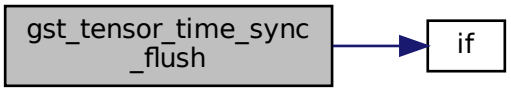
A function to be called while processing a flushing event. It should clear old buffer and reset pad data.

Parameters

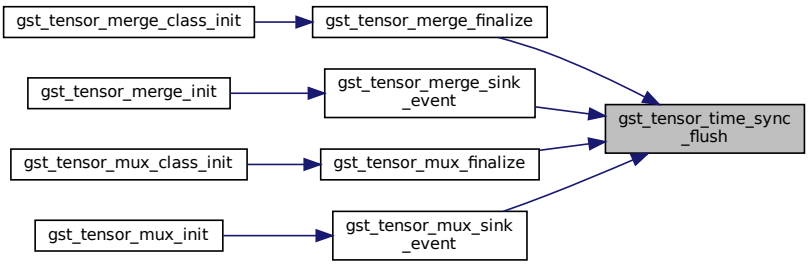
collect	Collect pad.

Definition at line 263 of file nnstreamer_plugin_api_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.93.3.35 gst_tensor_time_sync_get_current_time()

```
gboolean gst_tensor_time_sync_get_current_time (
            GstCollectPads * collect,
            tensor_time_sync_data * sync,
            GstClockTime * current_time,
            GstBuffer * tensors_buf )
```

A function call to decide current timestamp among collected pads based on PTS. It will decide current timestamp according to sync option. GstMeta is also copied with same sync mode.

Returns

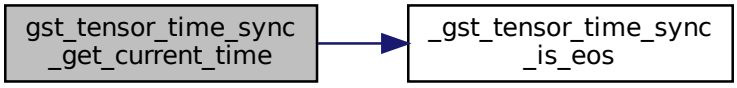
True / False. If EOS, it return TRUE.

Parameters

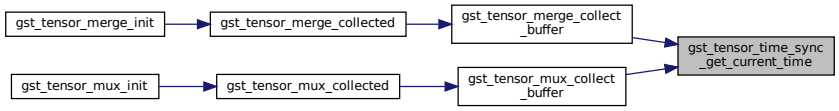
collect	Collect pad.
sync	Synchronization Option (NOSYNC, SLOWEST, BASEPAD, END)
current_time	Current time
tensors_buf	Generated GstBuffer for Collected Buffer

Definition at line 203 of file nnstreamer_plugin_api_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.93.3.36 gst_tensor_time_sync_get_mode()

```
tensor_time_sync_mode gst_tensor_time_sync_get_mode (
            const gchar * str )
```

Get the corresponding mode from the string value.

Parameters

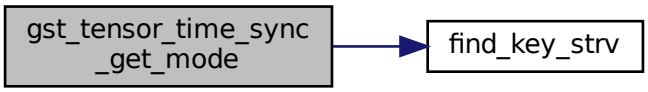
in	*str*	The string value for the mode.

Returns

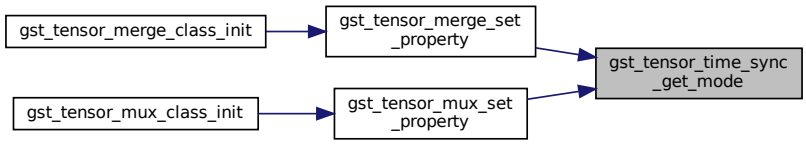
Corresponding mode for the string. SYNC_END for errors.

Definition at line 101 of file nnstreamer_plugin_api_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.93.3.37 gst_tensor_time_sync_get_mode_string()

```
const gchar* gst_tensor_time_sync_get_mode_string (
            tensor_time_sync_mode mode )
```

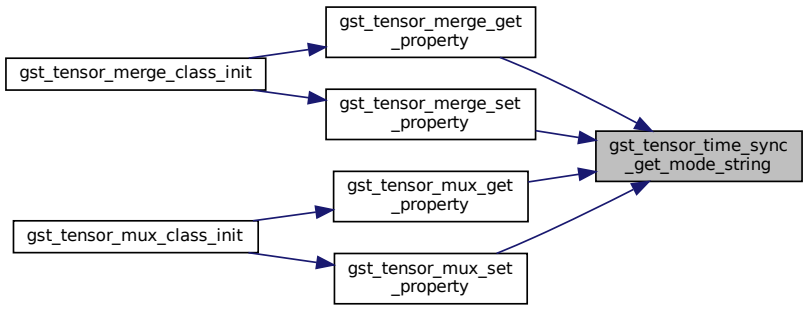
Get the time-sync mode string.

Returns

Corresponding mode string.

Definition at line 115 of file nnstreamer_plugin_api_impl.c.

Here is the caller graph for this function:



9.93.3.38 gst_tensor_time_sync_set_option_data()

```
gboolean gst_tensor_time_sync_set_option_data (
            tensor_time_sync_data * sync )
```

Setup time sync option.

Parameters

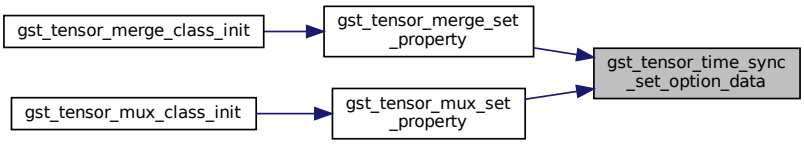
[in/out]	filter "this" pointer. Sync mode & option MUST BE set already.

Returns

True if successfully set the option.

Definition at line 126 of file nnstreamer_plugin_api_impl.c.

Here is the caller graph for this function:



9.93.3.39 gst_tensors_caps_from_config()

```
GstCaps* gst_tensors_caps_from_config (
            const GstTensorsConfig * config )
```

Get caps from tensors config (for other/tensors)

Parameters

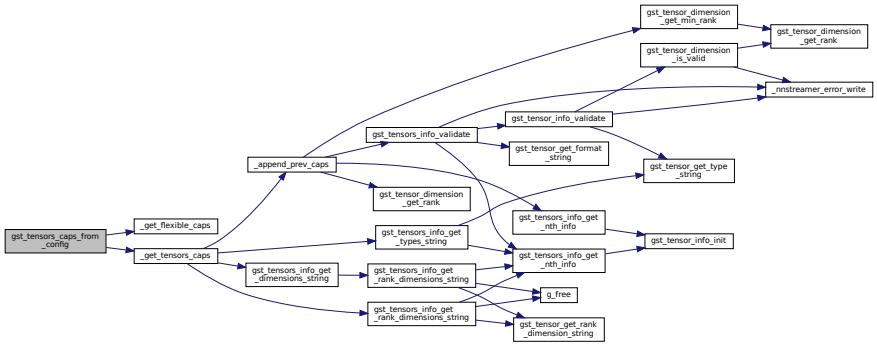
config	tensors config info

Returns

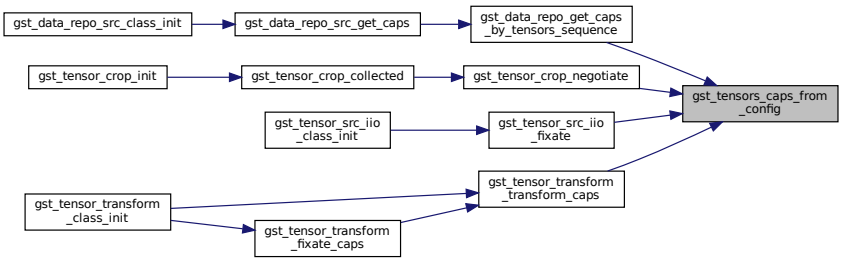
caps for given config

Definition at line 1372 of file nnstreamer_plugin_api_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.93.3.40 gst_tensors_config_from_peer()

```
gboolean gst_tensors_config_from_peer (
            GstPad * pad,
            GstTensorsConfig * config,
            gboolean * is_fixed )
```

Parse caps from peer pad and set tensors config.

Parameters

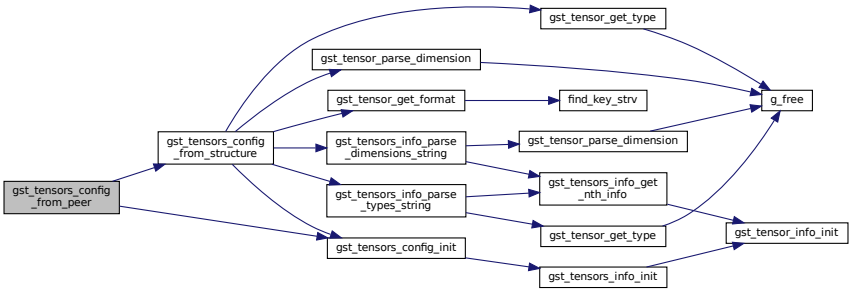
pad	GstPad to get the capabilities
config	tensors config structure to be filled
is_fixed	flag to be updated when peer caps is fixed (not mandatory, do nothing when the param is null)

Returns

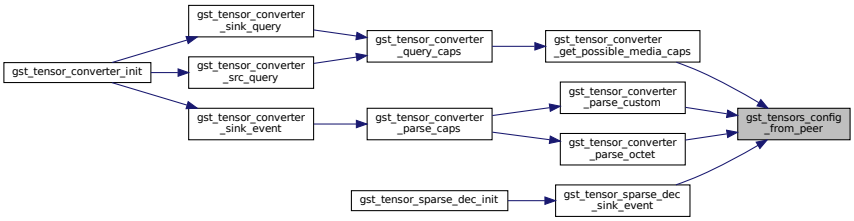
TRUE if successfully configured from peer

Definition at line 1041 of file nnstreamer_plugin_api_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.93.3.41 gst_tensors_config_from_structure()

```
gboolean gst_tensors_config_from_structure (
            GstTensorsConfig * config,
            const GstStructure * structure )
```

Parse structure and set tensors config (for other/tensors)

Parameters

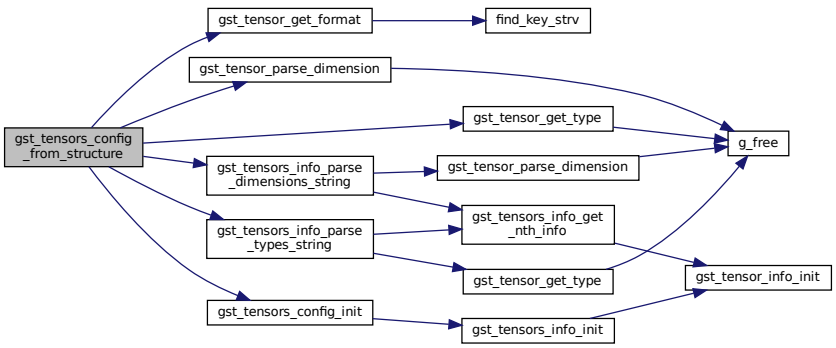
config	tensors config structure to be filled
structure	structure to be interpreted

Returns

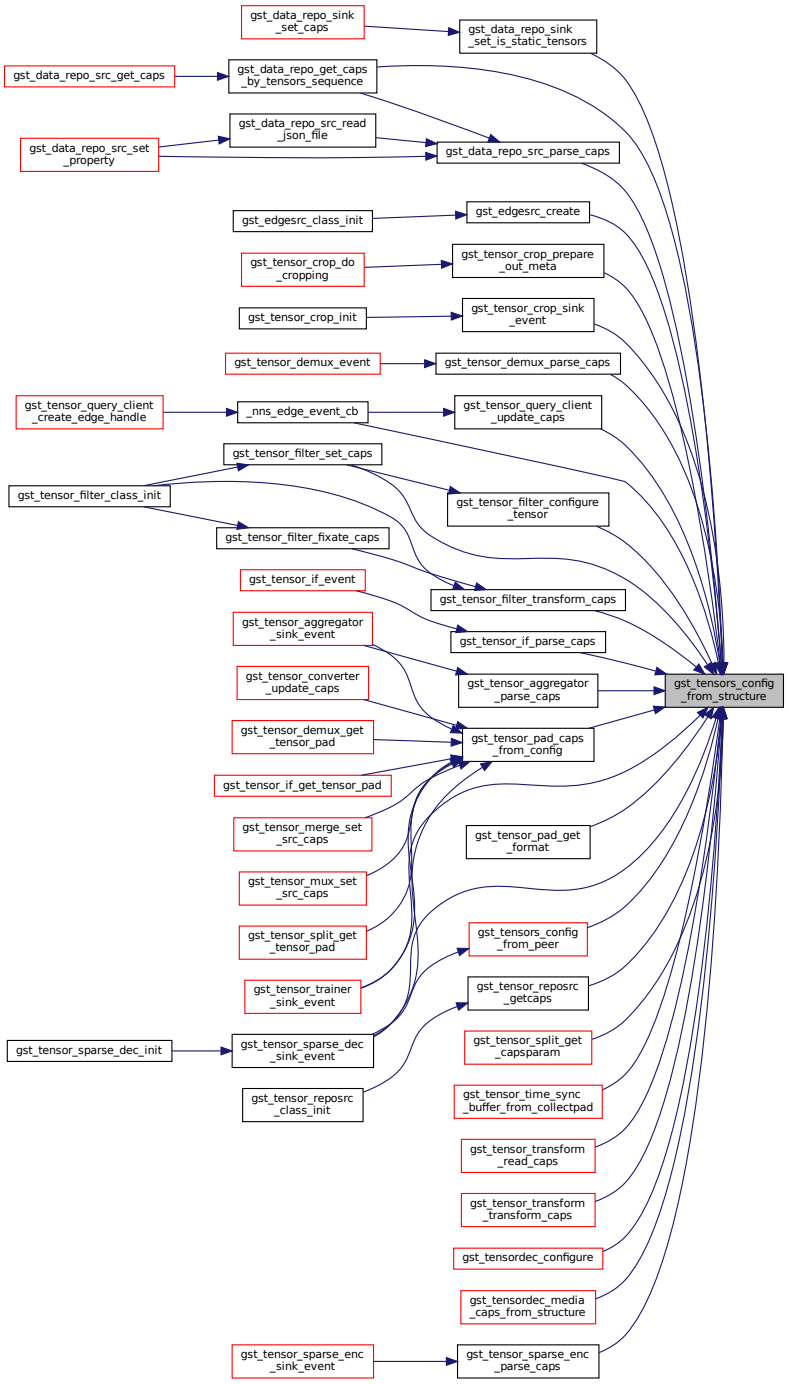
TRUE if no error

Definition at line 1413 of file nnstreamer_plugin_api_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.93.3.42 set_property_value()

```
static void set_property_value (
            GValue * prop_value,
```

```
const GParamSpec * param_spec,
const gchar * property_value )  [static]
```

Sets the value of a property based on the specified property value and GParamSpec.

Parameters

prop_value	A pointer to the GValue where the property value will be set.
param_spec	A pointer to the GParamSpec that describes the property.
property_value	A string representing the value to be set for the property.

Note

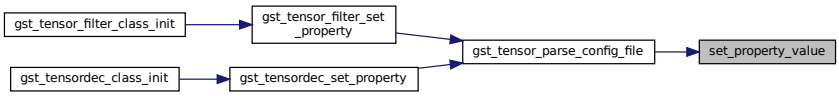
This API is intended to be used by gst_tensor_parse_config_file ()

default is string

Definition at line 1862 of file nnstreamer_plugin_api_impl.c.

Here is the call graph for this function:

Here is the caller graph for this function:



9.93.4 Variable Documentation

9.93.4.1 gst_tensor_time_sync_mode_string

```
const gchar* gst_tensor_time_sync_mode_string[]  [static]
```

Initial value:

```
= {
  [SYNC_NOSYNC] = "nosync",
  [SYNC_SLOWEST] = "slowest",
  [SYNC_BASEPAD] = "basepad",
  [SYNC_REFRESH] = "refresh",
  [SYNC_END] = NULL
}
```

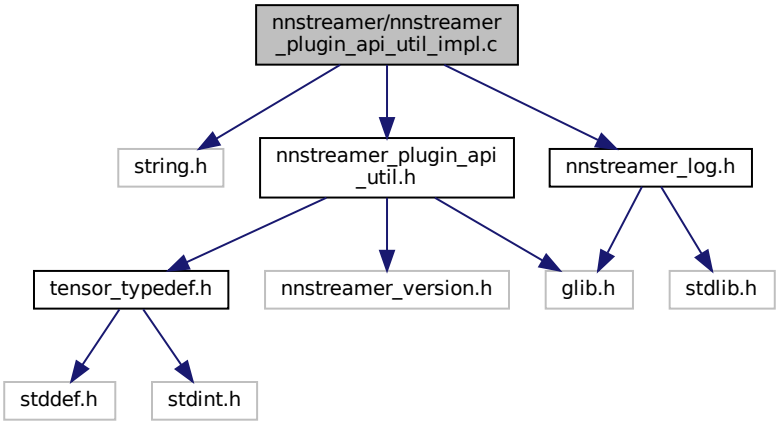
Definition at line 20 of file nnstreamer_plugin_api_impl.c.

9.94 nnstreamer/nnstreamer_plugin_api_util_impl.c File Reference

Tensor common util functions for NNStreamer. (No gst dependency)

```
#include <string.h>
#include "nnstreamer_plugin_api_util.h"
#include "nnstreamer_log.h"
```

Include dependency graph for nnstreamer_plugin_api_util_impl.c:



Macros

- #define GST_TENSOR_META_MAGIC (0xfeedcced)

  Magic number of tensor meta.
- #define GST_TENSOR_META_MAGIC_VALID(m) ((m) == GST_TENSOR_META_MAGIC)

  Macro to check the tensor meta.
- #define GST_TENSOR_META_VERSION_VALID(v) (((v) & 0xDE000000) == 0xDE000000)

  Macro to check the meta version.
- #define GST_TENSOR_META_MAKE_VERSION(major, minor) ((major) << 12 | (minor) | 0xDE000000)

  Macro to get the version of tensor meta.
- #define GST_TENSOR_META_VERSION GST_TENSOR_META_MAKE_VERSION(1,0)

  The version of tensor meta.
- #define GST_TENSOR_META_IS_V1(v) (GST_TENSOR_META_VERSION_VALID(v) && (((v) & 0x00FF↩F000) & GST_TENSOR_META_MAKE_VERSION(1,0)))

  Macro to check the version of tensor meta.
- #define GST_TENSOR_META_IS_VALID(m) ((m) && GST_TENSOR_META_MAGIC_VALID ((m)->magic) && GST_TENSOR_META_VERSION_VALID ((m)->version))

  Macro to check the meta is valid.

Functions

- static gint _gcd (gint a, gint b)

  Internal function, copied from gst_util_greatest_common_divisor() to remove dependency of gstreamer.
- static gint _compare_rate (gint a_n, gint a_d, gint b_n, gint b_d)

  Internal function, copied from gst_util_fraction_compare() to remove dependency of gstreamer.
- void gst_tensor_info_init (GstTensorInfo *info)

  Initialize the tensor info structure.
- void gst_tensor_info_free (GstTensorInfo *info)

  Free allocated data in tensor info structure.
- gsize gst_tensor_info_get_size (const GstTensorInfo *info)

  Get data size of single tensor.
- gboolean gst_tensor_info_validate (const GstTensorInfo *info)

  Check the tensor info is valid.
- gboolean gst_tensor_info_is_equal (const GstTensorInfo *i1, const GstTensorInfo *i2)

  Compare tensor info.
- void gst_tensor_info_copy_n (GstTensorInfo *dest, const GstTensorInfo *src, const guint n)

  Copy tensor info up to n elements.
- void gst_tensor_info_copy (GstTensorInfo *dest, const GstTensorInfo *src)

  Copy tensor info.
- gboolean gst_tensor_info_convert_to_meta (GstTensorInfo *info, GstTensorMetaInfo *meta)

  Convert GstTensorInfo structure to GstTensorMetaInfo.
- guint gst_tensor_info_get_rank (const GstTensorInfo *info)

  Get tensor rank.
- GstTensorInfo * gst_tensors_info_get_nth_info (GstTensorsInfo *info, guint index)

  Get the pointer of nth tensor information.
- void gst_tensors_info_init (GstTensorsInfo *info)

  Initialize the tensors info structure.
- void gst_tensors_info_free (GstTensorsInfo *info)

  Free allocated data in tensors info structure.
- gsize gst_tensors_info_get_size (const GstTensorsInfo *info, gint index)

  Get data size of single tensor.
- gboolean gst_tensors_info_validate (const GstTensorsInfo *info)

  Check the tensors info is valid.
- gboolean gst_tensors_info_is_equal (const GstTensorsInfo *i1, const GstTensorsInfo *i2)

  Compare tensors info.
- void gst_tensors_info_copy (GstTensorsInfo *dest, const GstTensorsInfo *src)

  Copy tensor info.
- guint gst_tensors_info_parse_dimensions_string (GstTensorsInfo *info, const gchar *dim_string)

  Parse the string of dimensions.
- guint gst_tensors_info_parse_types_string (GstTensorsInfo *info, const gchar *type_string)

  Parse the string of types.
- guint gst_tensors_info_parse_names_string (GstTensorsInfo *info, const gchar *name_string)

  Parse the string of names.
- gchar * gst_tensors_info_get_dimensions_string (const GstTensorsInfo *info)

  Get the string of dimensions in tensors info.
- gchar * gst_tensors_info_get_rank_dimensions_string (const GstTensorsInfo *info, const unsigned int rank)

  Get the string of dimensions in tensors info and rank count.
- gchar * gst_tensors_info_get_types_string (const GstTensorsInfo *info)

  Get the string of types in tensors info.
- gchar * gst_tensors_info_get_names_string (const GstTensorsInfo *info)

Get the string of tensor names in tensors info.

- gchar * gst_tensors_info_to_string (const GstTensorsInfo *info)

  GstTensorsInfo represented as a string. Caller should free it.
- void gst_tensors_config_init (GstTensorsConfig *config)

  Initialize the tensors config info structure (for other/tensors)
- void gst_tensors_config_free (GstTensorsConfig *config)

  Free allocated data in tensors config structure.
- gboolean gst_tensors_config_validate (const GstTensorsConfig *config)

  Check the tensors are all configured.
- gboolean gst_tensors_config_is_equal (const GstTensorsConfig *c1, const GstTensorsConfig *c2)

  Compare tensor config info.
- void gst_tensors_config_copy (GstTensorsConfig *dest, const GstTensorsConfig *src)

  Copy tensors config.
- gchar * gst_tensors_config_to_string (const GstTensorsConfig *config)

  Tensor config represented as a string. Caller should free it.
- gboolean gst_tensor_dimension_is_valid (const tensor_dim dim)

  Check the tensor dimension is valid.
- gboolean gst_tensor_dimension_is_equal (const tensor_dim dim1, const tensor_dim dim2)

  Compare the tensor dimension.
- guint gst_tensor_dimension_get_rank (const tensor_dim dim)

  Get the rank of tensor dimension.
- guint gst_tensor_dimension_get_min_rank (const tensor_dim dim)

  Get the minimum rank of tensor dimension.
- guint gst_tensor_parse_dimension (const gchar *dimstr, tensor_dim dim)

  Parse tensor dimension parameter string.
- gchar * gst_tensor_get_dimension_string (const tensor_dim dim)

  Get dimension string from given tensor dimension.
- gchar * gst_tensor_get_rank_dimension_string (const tensor_dim dim, const unsigned int rank)

  Get dimension string from given tensor dimension and rank count.
- gboolean gst_tensor_dimension_string_is_equal (const gchar *dimstr1, const gchar *dimstr2)

  Compare dimension strings.
- gulong gst_tensor_get_element_count (const tensor_dim dim)

  Count the number of elements of a tensor.
- gsize gst_tensor_get_element_size (tensor_type type)

  Get element size of tensor type (byte per element)
- tensor_type gst_tensor_get_type (const gchar *typestr)

  Get tensor type from string input.
- const gchar * gst_tensor_get_type_string (tensor_type type)

  Get type string of tensor type.
- tensor_format gst_tensor_get_format (const gchar *format_str)

  Get tensor format from string input.
- const gchar * gst_tensor_get_format_string (tensor_format format)

  Get tensor format string.
- void gst_tensor_meta_info_init (GstTensorMetaInfo *meta)

  Initialize the tensor meta info structure.
- void gst_tensor_meta_info_get_version (GstTensorMetaInfo *meta, guint *major, guint *minor)

  Get the version of tensor meta.
- gboolean gst_tensor_meta_info_validate (GstTensorMetaInfo *meta)

  Check the meta info is valid.
- gsize gst_tensor_meta_info_get_header_size (GstTensorMetaInfo *meta)

  Get the header size to handle a tensor meta.

- gsize gst_tensor_meta_info_get_data_size (GstTensorMetaInfo *meta)

  Get the data size calculated from tensor meta.
- gboolean gst_tensor_meta_info_update_header (GstTensorMetaInfo *meta, gpointer header)

  Update header from tensor meta.
- gboolean gst_tensor_meta_info_parse_header (GstTensorMetaInfo *meta, gpointer header)

  Parse header and fill the tensor meta.
- gboolean gst_tensor_meta_info_convert (GstTensorMetaInfo *meta, GstTensorInfo *info)

  Convert GstTensorMetaInfo structure to GstTensorInfo.
- gint find_key_strv (const gchar **strv, const gchar *key)

  Find the index value of the given key string array.
- gchar * nnstreamer_version_string (void)

  Get the version of NNStreamer (string).
- void nnstreamer_version_fetch (guint *major, guint *minor, guint *micro)

  Get the version of NNStreamer (int, divided).

Variables

- static const gchar * tensor_element_typename [ ]

  String representations for each tensor element type.
- static const guint tensor_element_size [ ]

  Byte-per-element of each tensor element type.
- static const gchar * tensor_format_name [ ]

  String representations for tensor format.

9.94.1 Detailed Description

Tensor common util functions for NNStreamer. (No gst dependency)

Date

28 Jan 2022

See also

https://github.com/nnstreamer/nnstreamer

Author

Gichan Jang gichan2.jang@samsung.com

Bug No known bugs except for NYI items

9.94.2 Macro Definition Documentation

9.94.2.1 GST_TENSOR_META_IS_V1

```
#define GST_TENSOR_META_IS_V1(
              v ) (GST_TENSOR_META_VERSION_VALID(v) && (((v) & 0x00FFF000) & GST_TENSOR_META_MAKE_VERSION(1,0)
```

Macro to check the version of tensor meta.

Definition at line 1360 of file nnstreamer_plugin_api_util_impl.c.

9.94.2.2 GST_TENSOR_META_IS_VALID

```
#define GST_TENSOR_META_IS_VALID(
              m ) ((m) && GST_TENSOR_META_MAGIC_VALID ((m)->magic) && GST_TENSOR_META_VERSION_VALID
((m)->version))
```

Macro to check the meta is valid.

Definition at line 1365 of file nnstreamer_plugin_api_util_impl.c.

9.94.2.3 GST_TENSOR_META_MAGIC

```
#define GST_TENSOR_META_MAGIC (0xfeedcced)
```

Magic number of tensor meta.

Definition at line 1335 of file nnstreamer_plugin_api_util_impl.c.

9.94.2.4 GST_TENSOR_META_MAGIC_VALID

```
#define GST_TENSOR_META_MAGIC_VALID(
              m ) ((m) == GST_TENSOR_META_MAGIC)
```

Macro to check the tensor meta.

Definition at line 1340 of file nnstreamer_plugin_api_util_impl.c.

9.94.2.5 GST_TENSOR_META_MAKE_VERSION

```
#define GST_TENSOR_META_MAKE_VERSION(
              major,
              minor ) ((major) << 12 | (minor) | 0xDE000000)
```

Macro to get the version of tensor meta.

Definition at line 1350 of file nnstreamer_plugin_api_util_impl.c.

9.94.2.6 GST_TENSOR_META_VERSION

```
#define GST_TENSOR_META_VERSION GST_TENSOR_META_MAKE_VERSION(1,0)
```

The version of tensor meta.

Definition at line 1355 of file nnstreamer_plugin_api_util_impl.c.

9.94.2.7 GST_TENSOR_META_VERSION_VALID

```
#define GST_TENSOR_META_VERSION_VALID(
              v ) (((v) & 0xDE000000) == 0xDE000000)
```

Macro to check the meta version.

Definition at line 1345 of file nnstreamer_plugin_api_util_impl.c.

9.94.3 Function Documentation

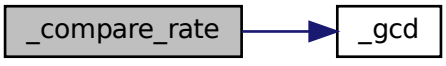
9.94.3.1 _compare_rate()

```
static gint _compare_rate (
             gint a_n,
             gint a_d,
             gint b_n,
             gint b_d )  [static]
```

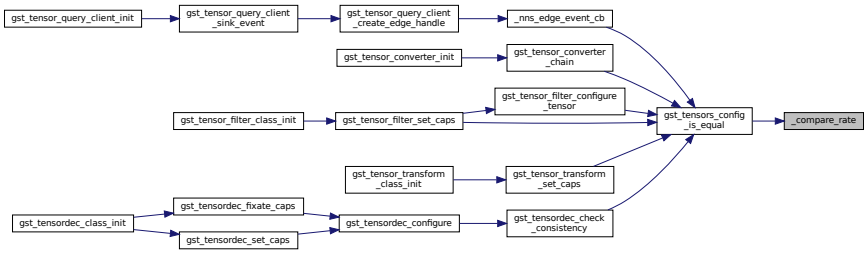
Internal function, copied from gst_util_fraction_compare() to remove dependency of gstreamer.

Definition at line 83 of file nnstreamer_plugin_api_util_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



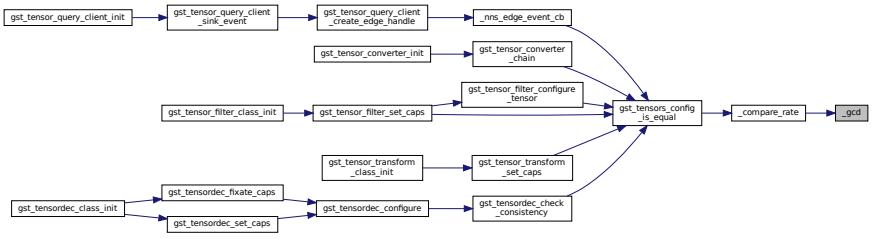
9.94.3.2 _gcd()

```
static gint _gcd (
            gint a,
            gint b )  [static]
```

Internal function, copied from gst_util_greatest_common_divisor() to remove dependency of gstreamer.

Definition at line 67 of file nnstreamer_plugin_api_util_impl.c.

Here is the caller graph for this function:



9.94.3.3 find_key_strv()

```
gint find_key_strv (
            const gchar ** strv,
            const gchar * key )
```

Find the index value of the given key string array.

Returns

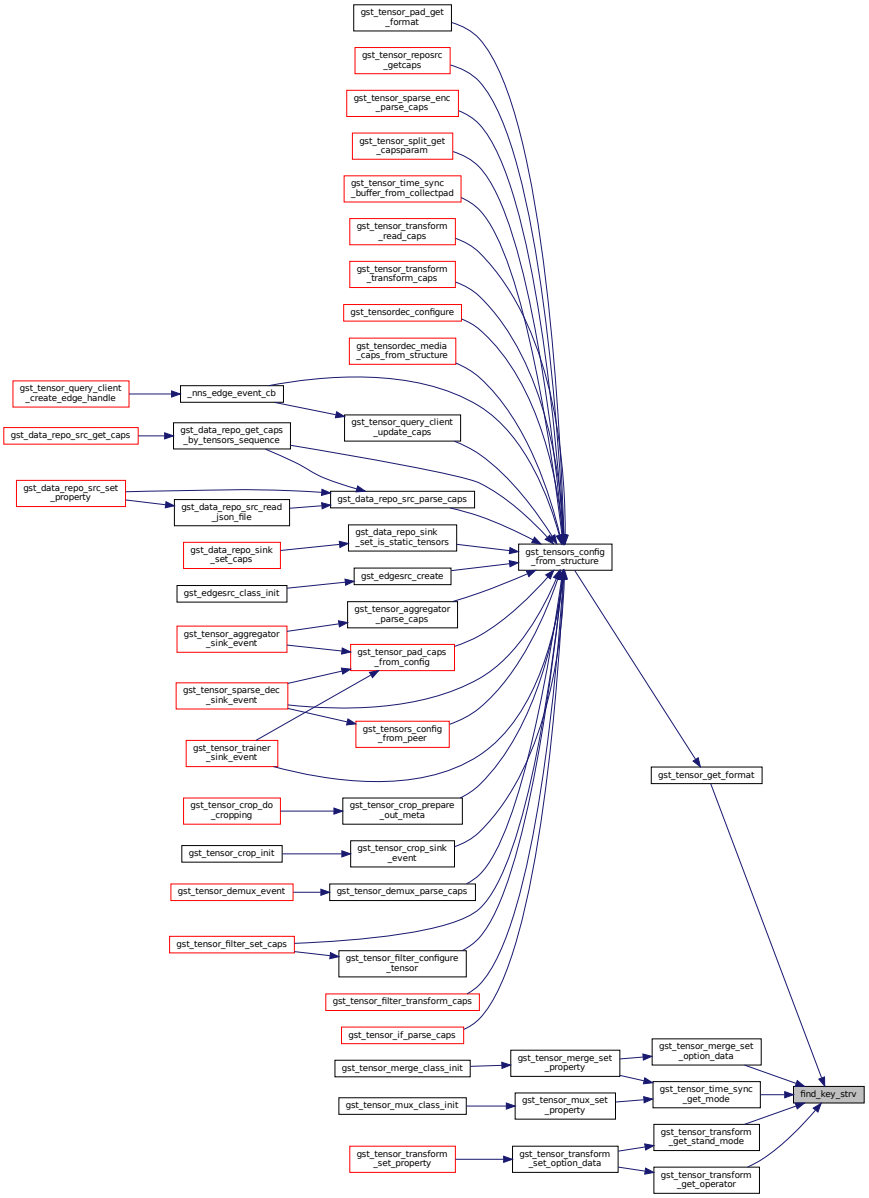
Corresponding index. Returns -1 if not found.

Parameters

strv	Null terminated array of gchar *
key	The key string value

Definition at line 1586 of file nnstreamer_plugin_api_util_impl.c.

Here is the caller graph for this function:



9.94.3.4 gst_tensor_dimension_get_min_rank()

```
guint gst_tensor_dimension_get_min_rank (
            const tensor_dim dim )
```

Get the minimum rank of tensor dimension.

The C-arrays with dim 4:4:4 and 4:4:4:1 have same data. In this case, this function returns min rank 3.

Parameters

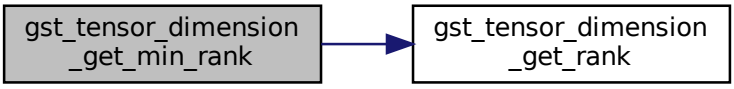
dim	tensor dimension.

Returns

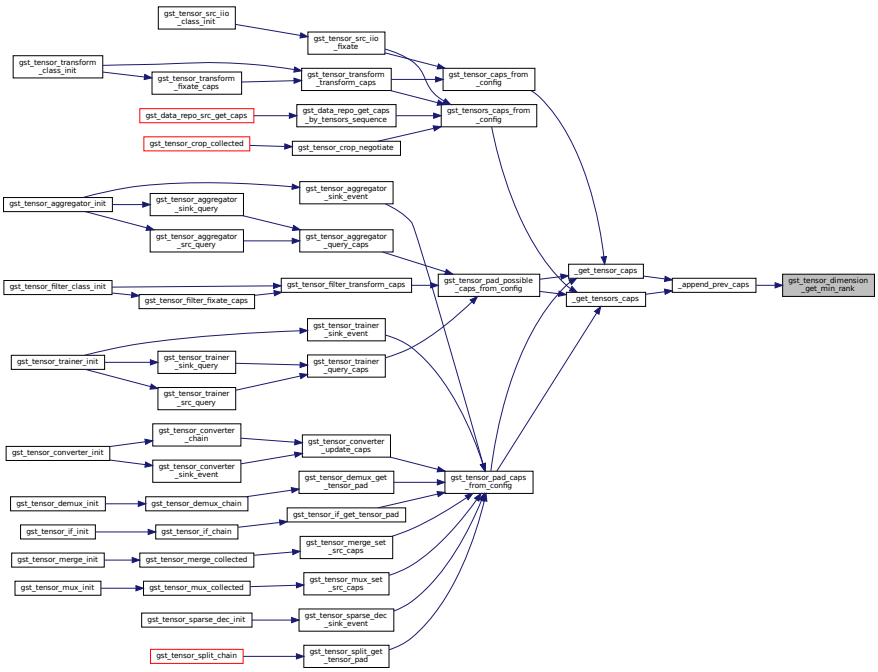
tensor rank (Minimum rank is 1 if given dimension is valid)

Definition at line 1017 of file nnstreamer_plugin_api_util_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.94.3.5 gst_tensor_dimension_get_rank()

```
guint gst_tensor_dimension_get_rank (
            const tensor_dim dim )
```

Get the rank of tensor dimension.

Parameters

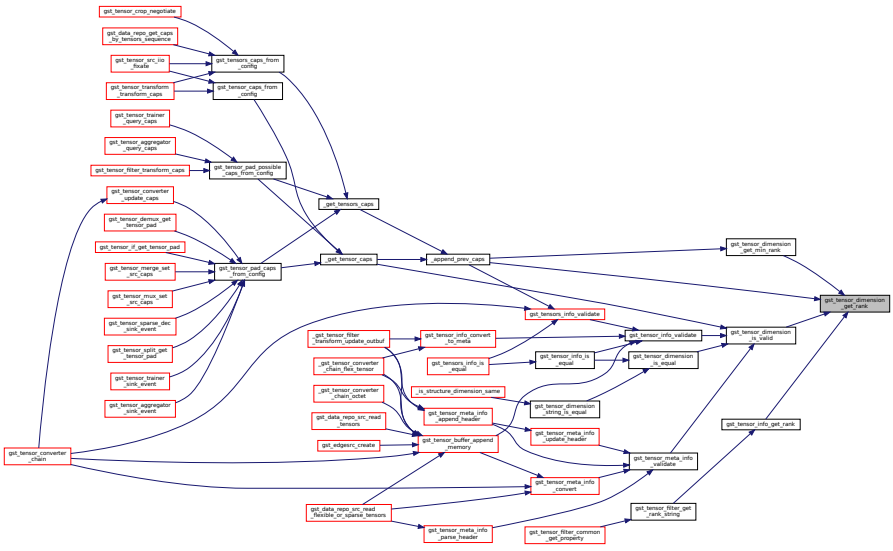
dim	tensor dimension.

Returns

tensor rank (Minimum rank is 1 if given dimension is valid)

Definition at line 998 of file nnstreamer_plugin_api_util_impl.c.

Here is the caller graph for this function:



9.94.3.6 gst_tensor_dimension_is_equal()

```
gboolean gst_tensor_dimension_is_equal (
            const tensor_dim dim1,
            const tensor_dim dim2 )
```

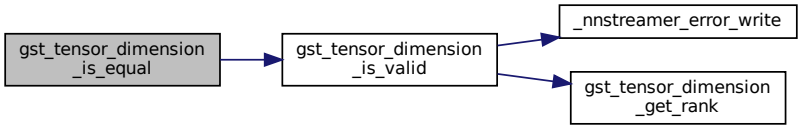
Compare the tensor dimension.

Returns

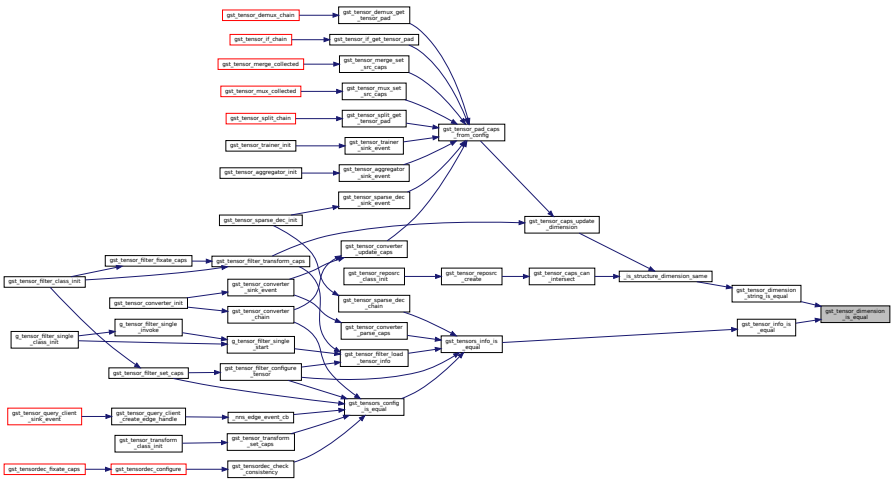
TRUE if given tensors have same dimension.

Definition at line 972 of file nnstreamer_plugin_api_util_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.94.3.7 gst_tensor_dimension_is_valid()

```
gboolean gst_tensor_dimension_is_valid (
            const tensor_dim dim )
```

Check the tensor dimension is valid.

Parameters

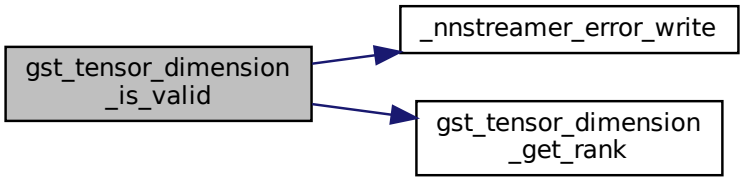
dim	tensor dimension

Returns

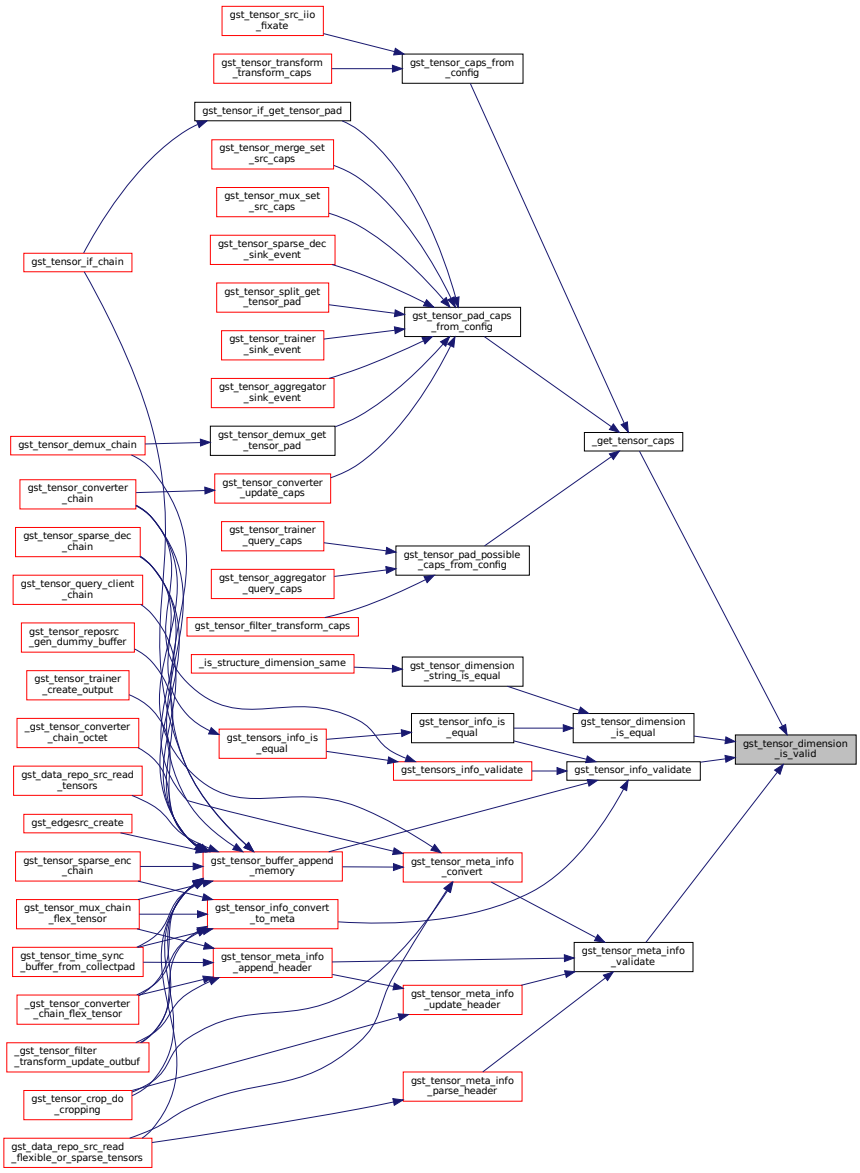
TRUE if dimension is valid

Definition at line 940 of file nnstreamer_plugin_api_util_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.94.3.8 gst_tensor_dimension_string_is_equal()

```
gboolean gst_tensor_dimension_string_is_equal (
            const gchar * dimstr1,
            const gchar * dimstr2 )
```

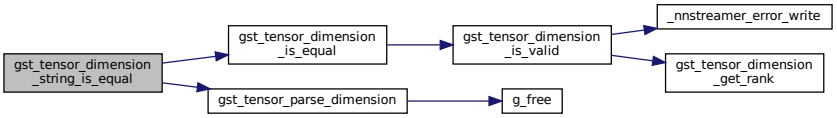
Compare dimension strings.

Returns

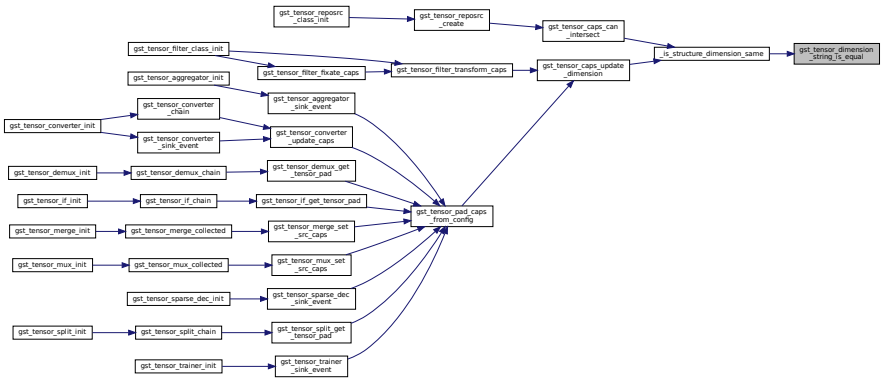
TRUE if equal, FALSE if given dimension strings are invalid or not equal.

Definition at line 1140 of file nnstreamer_plugin_api_util_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.94.3.9 gst_tensor_get_dimension_string()

```
gchar* gst_tensor_get_dimension_string (
            const tensor_dim dim )
```

Get dimension string from given tensor dimension.

Parameters

dim	tensor dimension

Returns

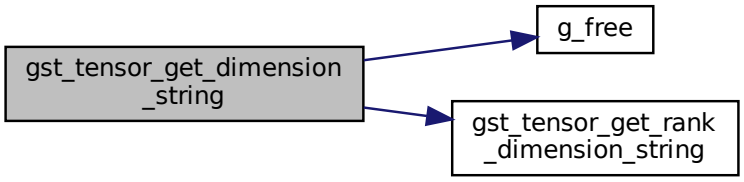
Formatted string of given dimension (d1:d2:d3:...:d15:d16).

Note

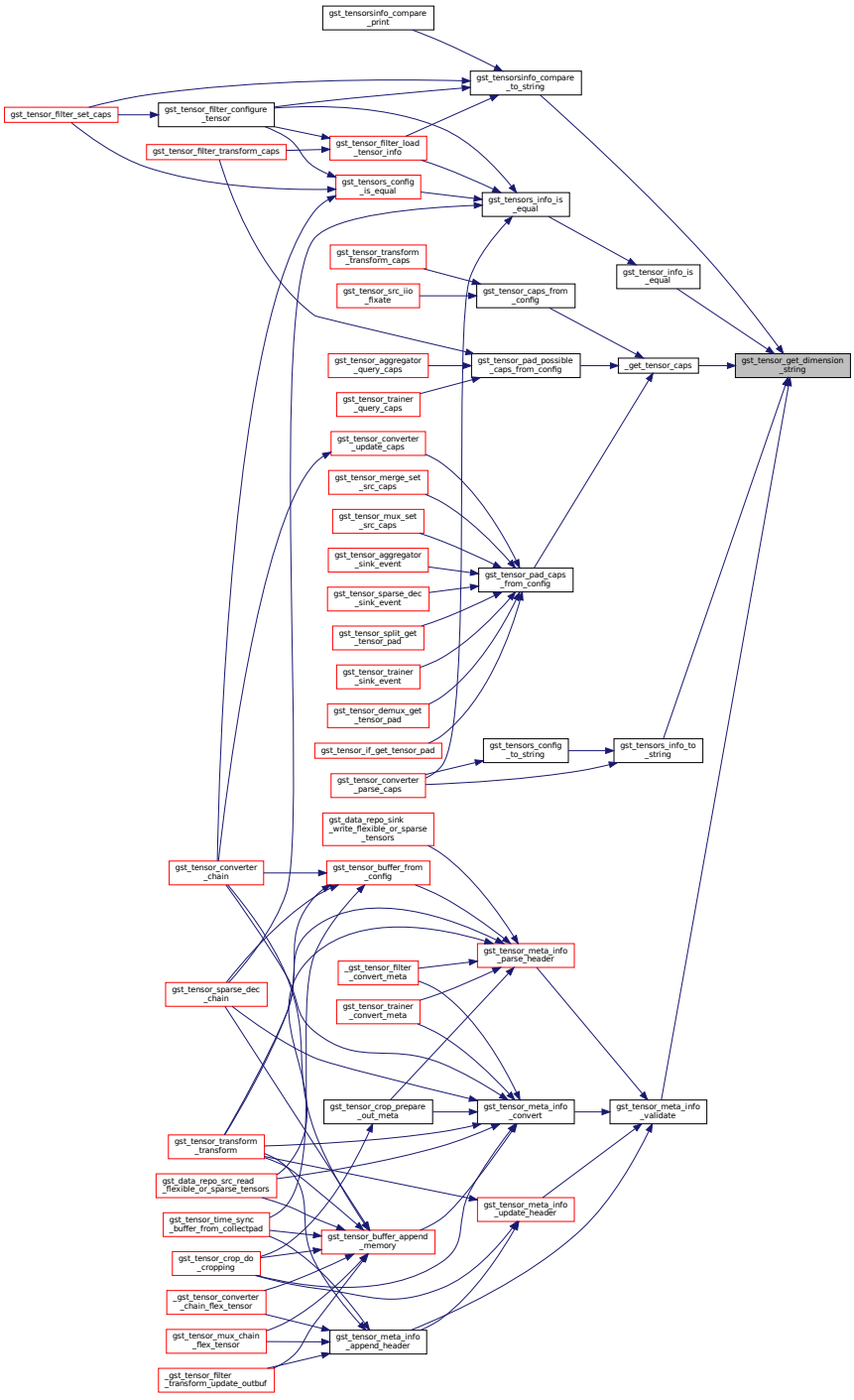
The returned value should be freed with g_free()

Definition at line 1083 of file nnstreamer_plugin_api_util_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.94.3.10 gst_tensor_get_element_count()

```
gulong gst_tensor_get_element_count (
            const tensor_dim dim )
```

Count the number of elements of a tensor.

Returns

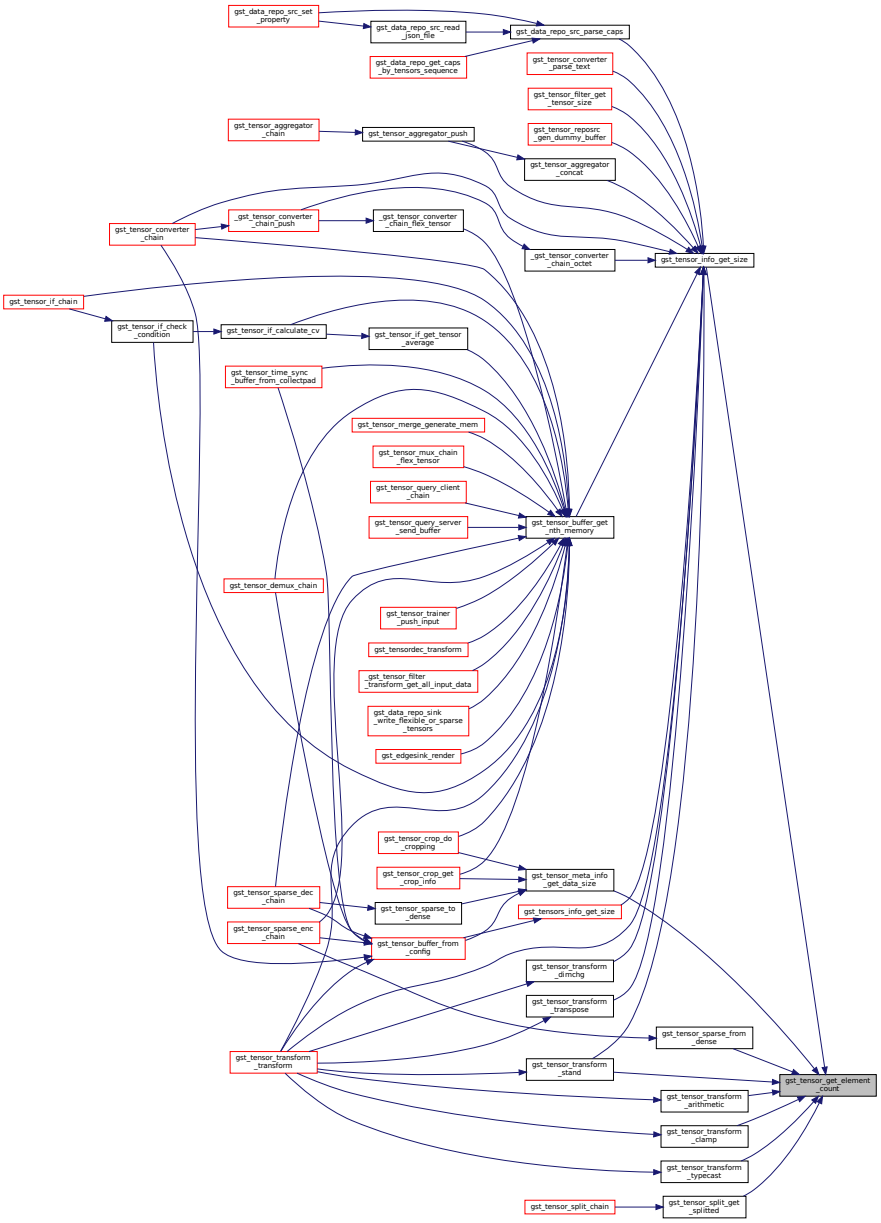
The number of elements. 0 if error.

Parameters

dim	The tensor dimension

Definition at line 1186 of file nnstreamer_plugin_api_util_impl.c.

Here is the caller graph for this function:



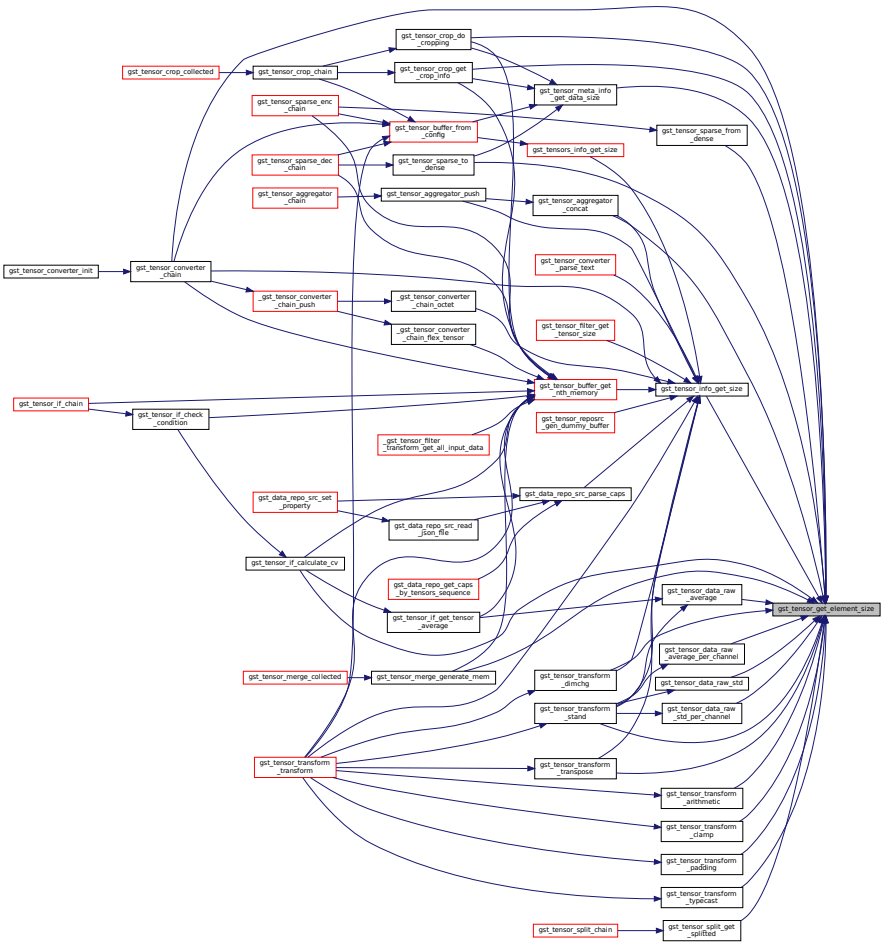
9.94.3.11 gst_tensor_get_element_size()

```
gsize gst_tensor_get_element_size (
            tensor_type type )
```

Get element size of tensor type (byte per element)

Definition at line 1205 of file nnstreamer_plugin_api_util_impl.c.

Here is the caller graph for this function:



9.94.3.12 gst_tensor_get_format()

```
tensor_format gst_tensor_get_format (
            const gchar * format_str )
```

Get tensor format from string input.

Parameters

format_str	The string format name, supposed to be one of tensor_format_name[].

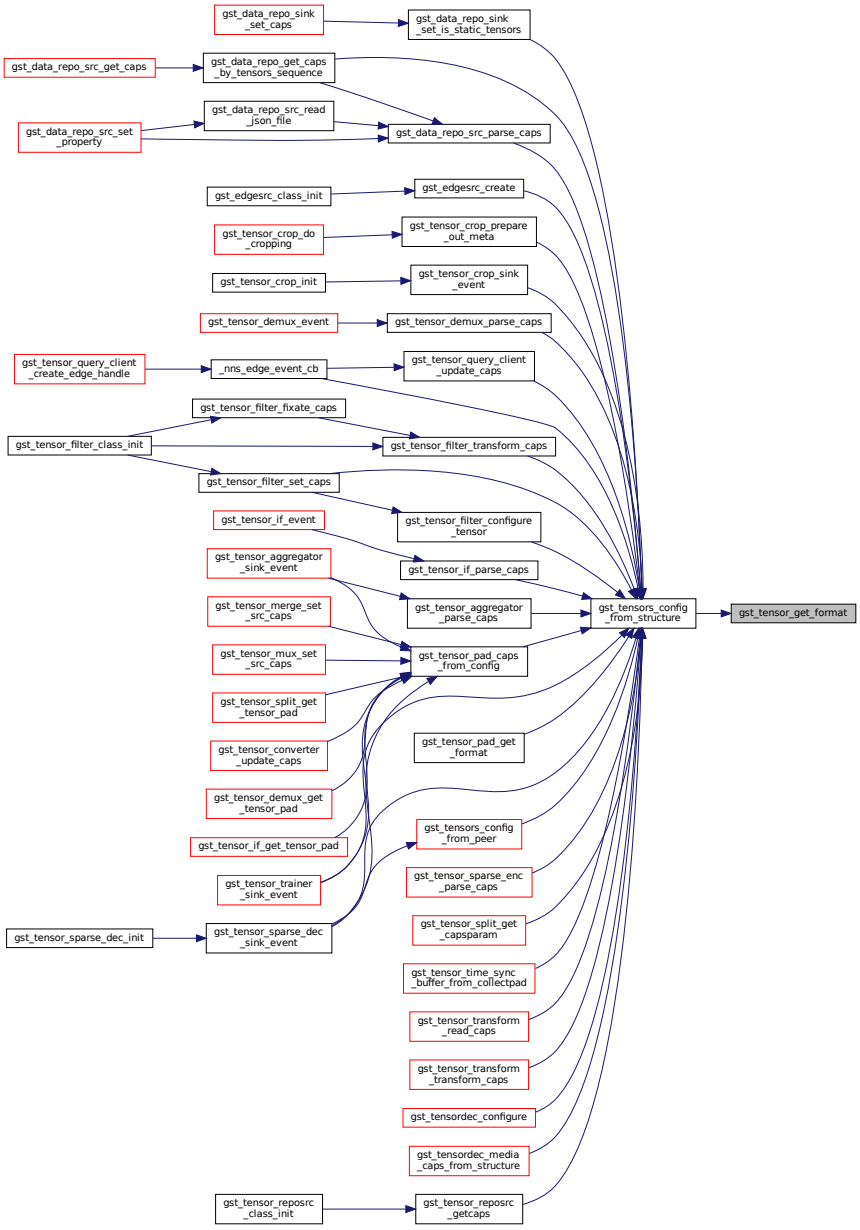
Returns

Corresponding tensor_format. _NNS_TENSOR_FORMAT_END if unrecognized value is there.

Definition at line 1309 of file nnstreamer_plugin_api_util_impl.c.

Here is the call graph for this function:

Here is the caller graph for this function:



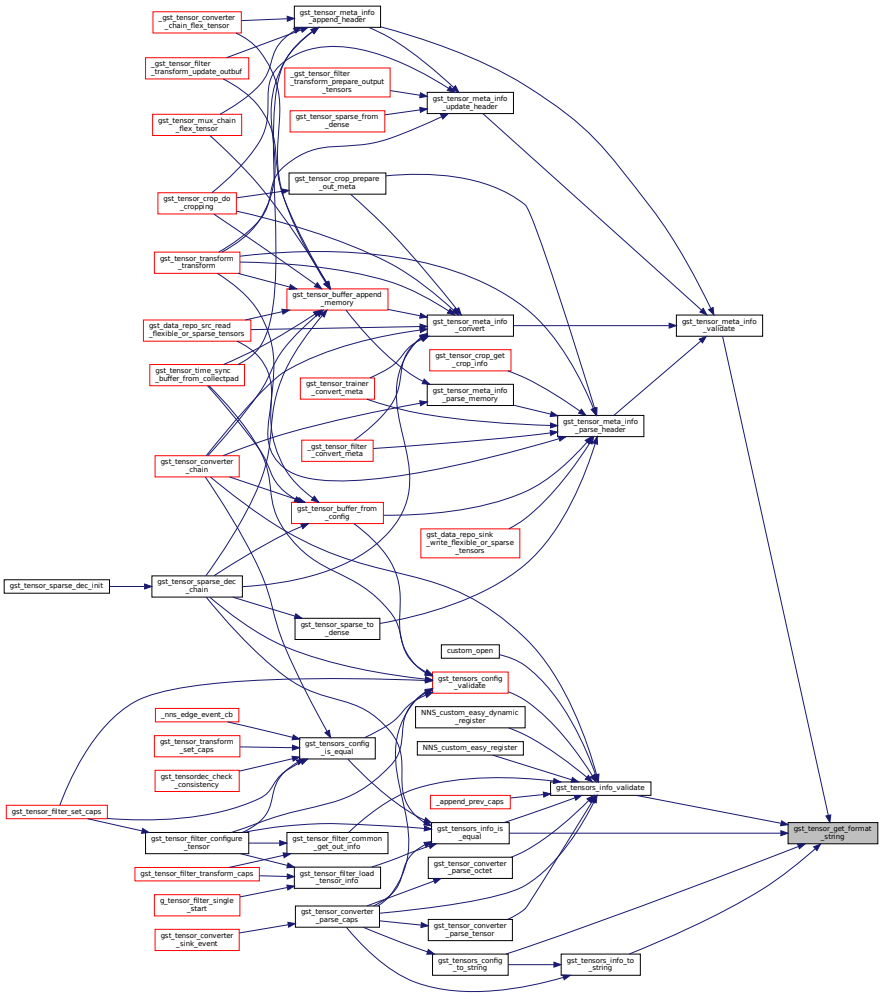
9.94.3.13 gst_tensor_get_format_string()

```
const gchar* gst_tensor_get_format_string (
            tensor_format format )
```

Get tensor format string.

Definition at line 1325 of file nnstreamer_plugin_api_util_impl.c.

Here is the caller graph for this function:



9.94.3.14 gst_tensor_get_rank_dimension_string()

```
gchar* gst_tensor_get_rank_dimension_string (
            const tensor_dim dim,
            const unsigned int rank )
```

Get dimension string from given tensor dimension and rank count.

Parameters

dim	tensor dimension
rank	rank count of given tensor dimension

Returns

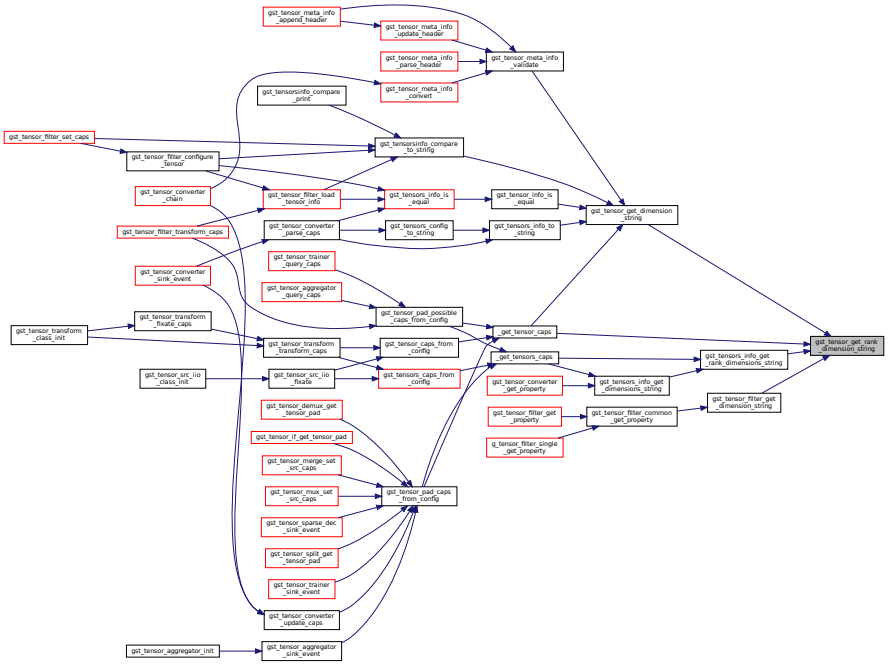
Formatted string of given dimension

Note

If rank count is 3, then returned string is 'd1:d2:d3`. The returned value should be freed with g_free().

Definition at line 1107 of file nnstreamer_plugin_api_util_impl.c.

Here is the caller graph for this function:



9.94.3.15 gst_tensor_get_type()

```
tensor_type gst_tensor_get_type (
            const gchar * typestr )
```

Get tensor type from string input.

Returns

Corresponding tensor_type. _NNS_END if unrecognized value is there.

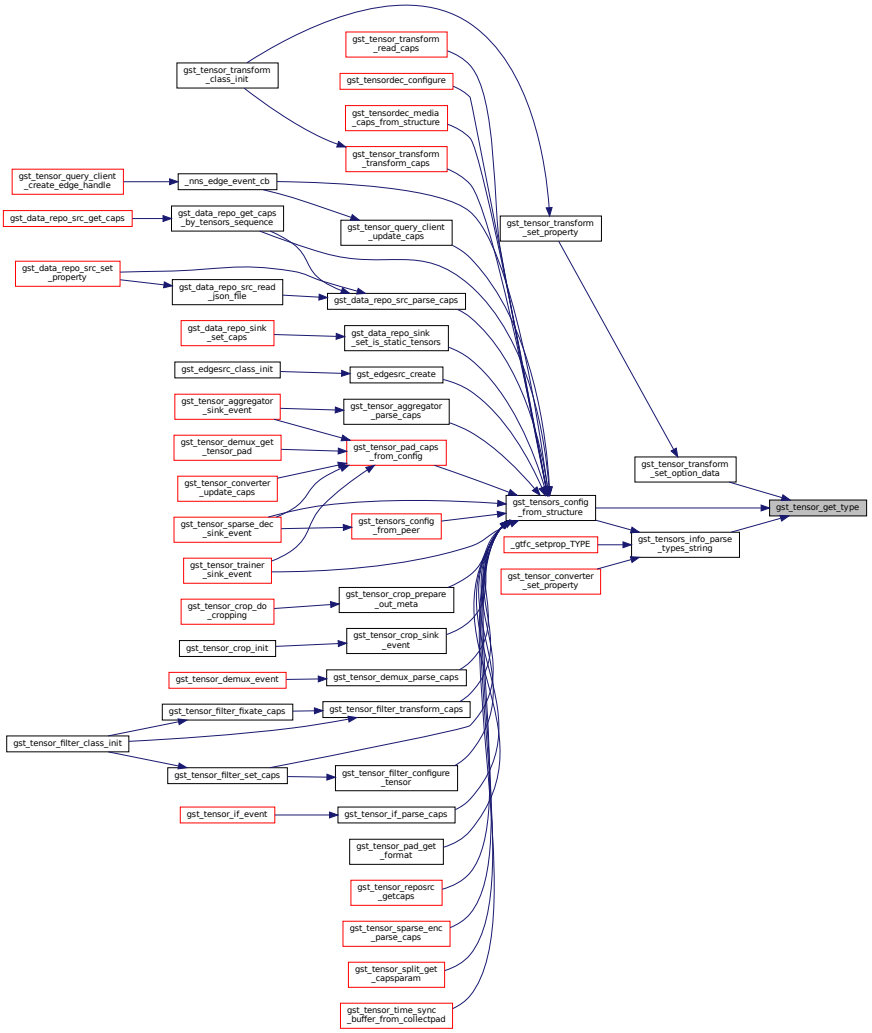
Parameters

typestr	The string type name, supposed to be one of tensor_element_typename[]

Definition at line 1218 of file nnstreamer_plugin_api_util_impl.c.

Here is the call graph for this function:

Here is the caller graph for this function:



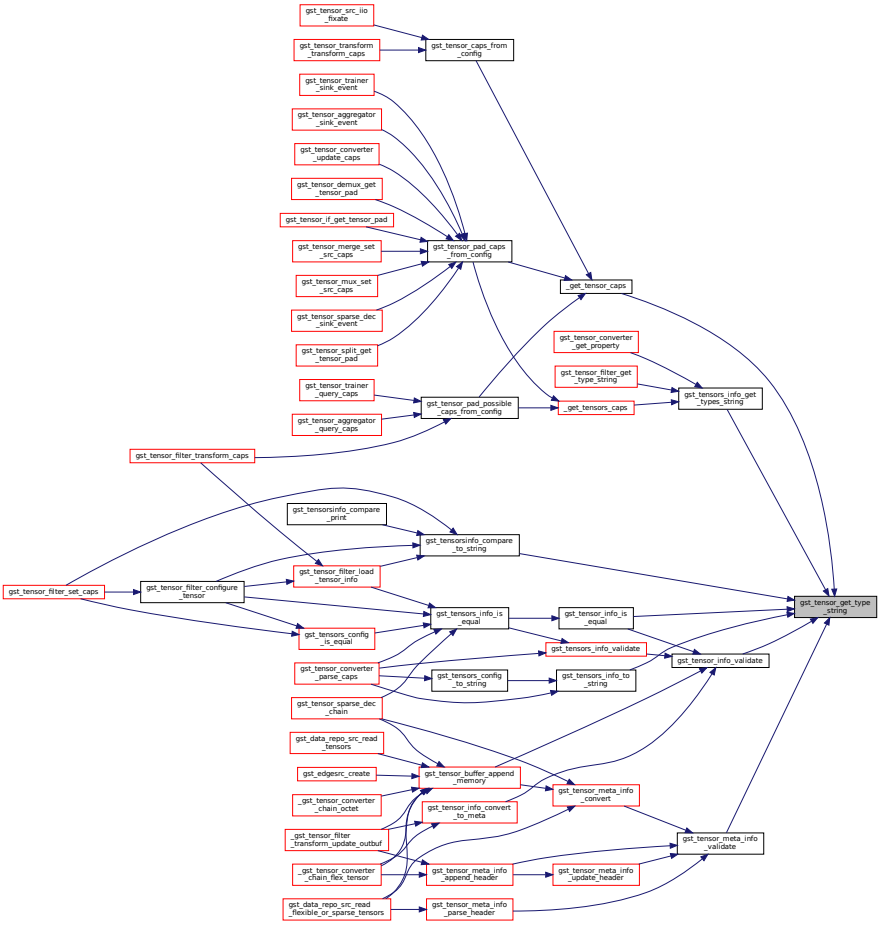
9.94.3.16 gst_tensor_get_type_string()

```
const gchar* gst_tensor_get_type_string (
            tensor_type type )
```

Get type string of tensor type.

Definition at line 1296 of file nnstreamer_plugin_api_util_impl.c.

Here is the caller graph for this function:



9.94.3.17 gst_tensor_info_convert_to_meta()

```
gboolean gst_tensor_info_convert_to_meta (
            GstTensorInfo * info,
            GstTensorMetaInfo * meta )
```

Convert GstTensorInfo structure to GstTensorMetaInfo.

Parameters

in	*info*	GstTensorInfo to be converted
out	*meta*	tensor meta structure to be filled

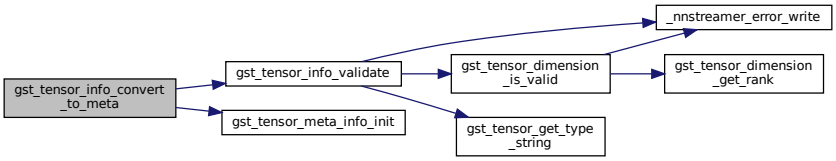
Returns

TRUE if successfully set the meta

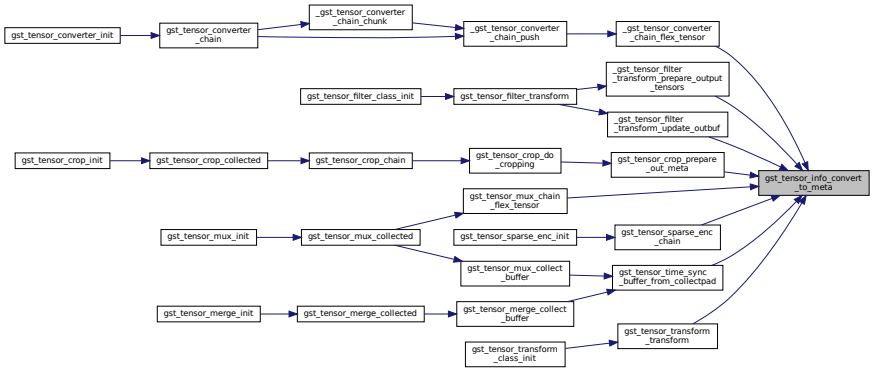
Todo handle rank from info.dimension

Definition at line 260 of file nnstreamer_plugin_api_util_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.94.3.18 gst_tensor_info_copy()

```
void gst_tensor_info_copy (
            GstTensorInfo * dest,
            const GstTensorInfo * src )
```

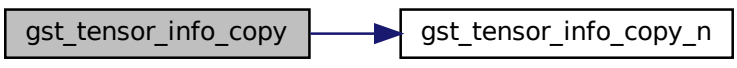
Copy tensor info.

Note

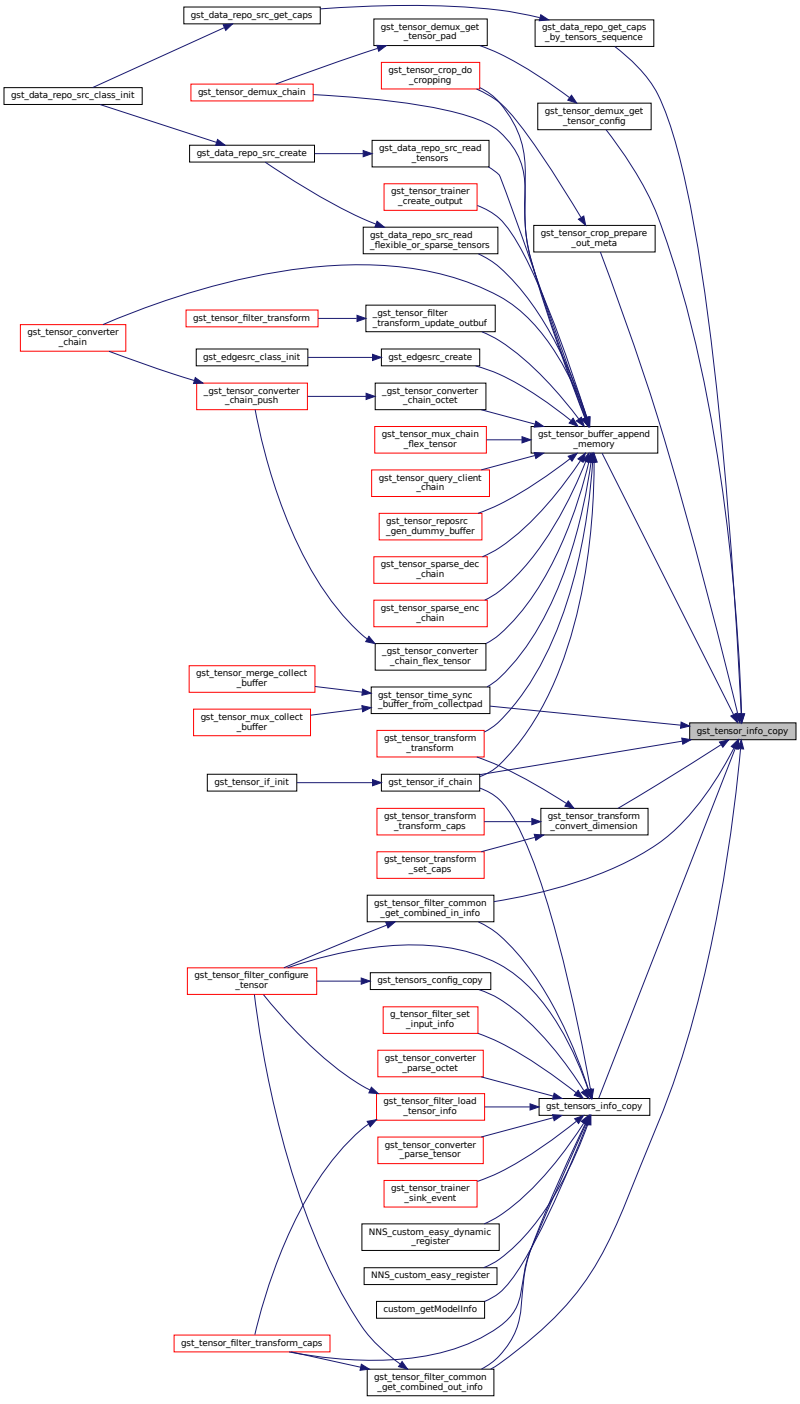
Copied info should be freed with gst_tensor_info_free()

Definition at line 248 of file nnstreamer_plugin_api_util_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.94.3.19 gst_tensor_info_copy_n()

```
void gst_tensor_info_copy_n (
            GstTensorInfo * dest,
```

```
const GstTensorInfo * src,
const guint n )
```

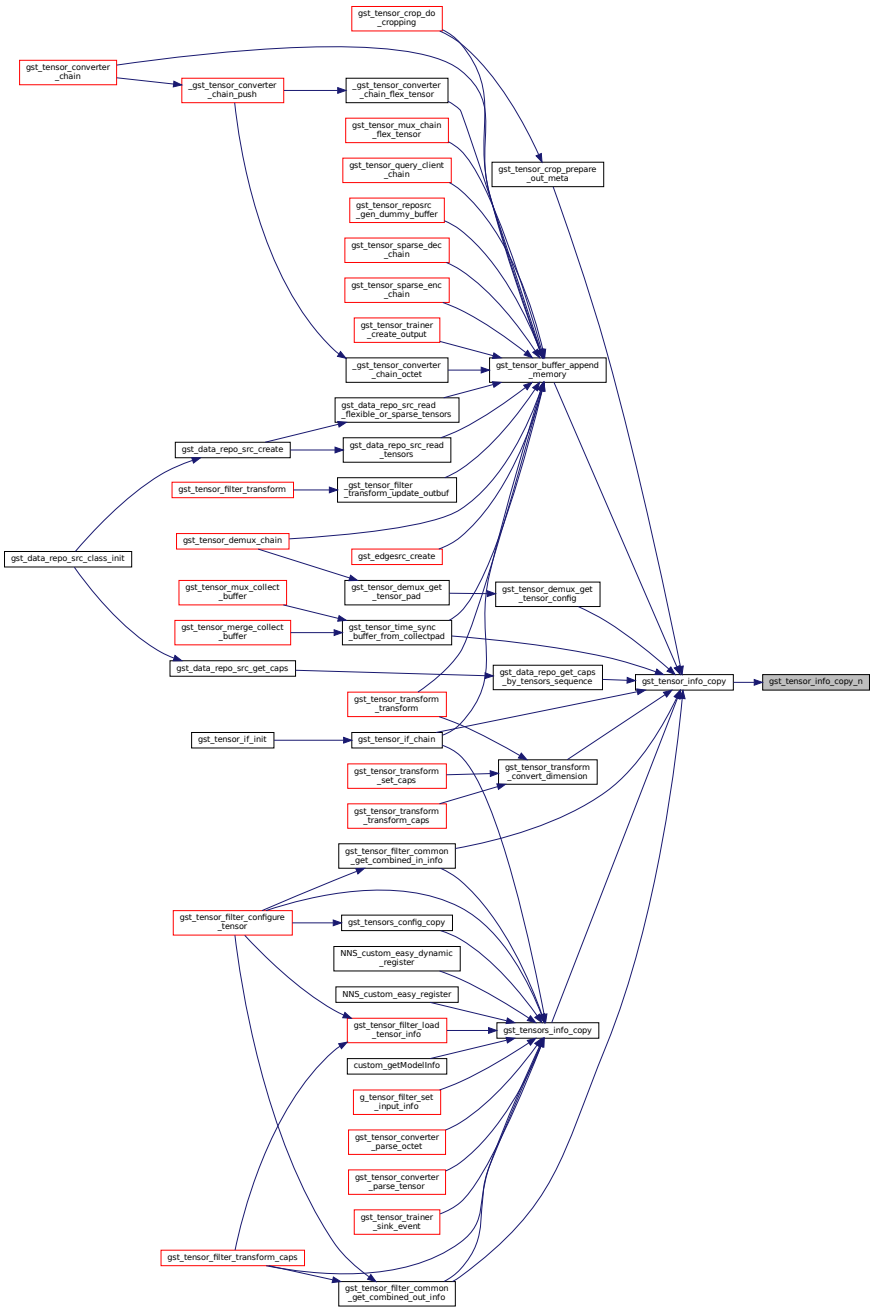
Copy tensor info up to n elements.

Note

Copied info should be freed with gst_tensor_info_free()

Definition at line 227 of file nnstreamer_plugin_api_util_impl.c.

Here is the caller graph for this function:



9.94.3.20 gst_tensor_info_free()

```
void gst_tensor_info_free (
            GstTensorInfo * info )
```

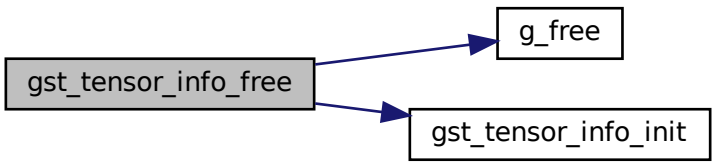
Free allocated data in tensor info structure.

Parameters

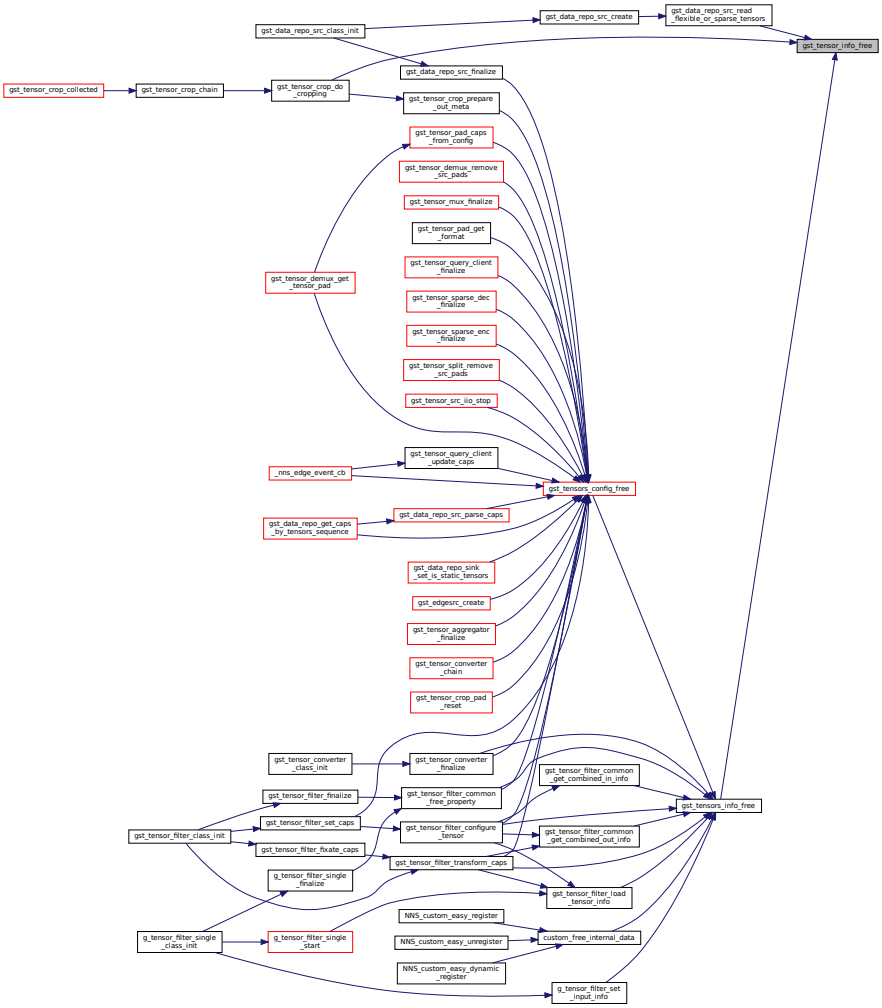
info	tensor info structure

Definition at line 140 of file nnstreamer_plugin_api_util_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.94.3.21 gst_tensor_info_get_rank()

```
guint gst_tensor_info_get_rank (
            const GstTensorInfo * info )
```

Get tensor rank.

Parameters

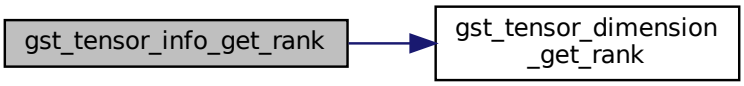
info	tensor info structure

Returns

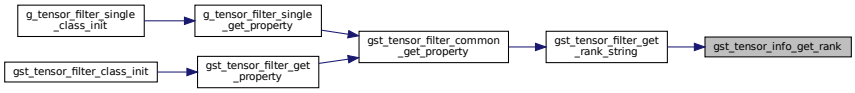
tensor rank (Minimum rank is 1 if given info is valid)

Definition at line 285 of file nnstreamer_plugin_api_util_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.94.3.22 gst_tensor_info_get_size()

```
gsize gst_tensor_info_get_size (
            const GstTensorInfo * info )
```

Get data size of single tensor.

Parameters

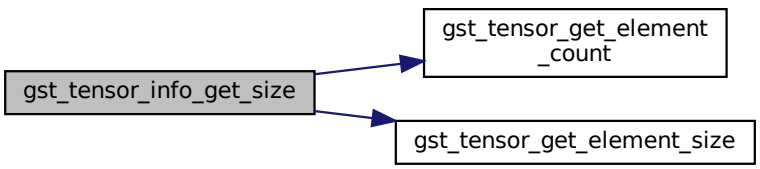
info	tensor info structure

Returns

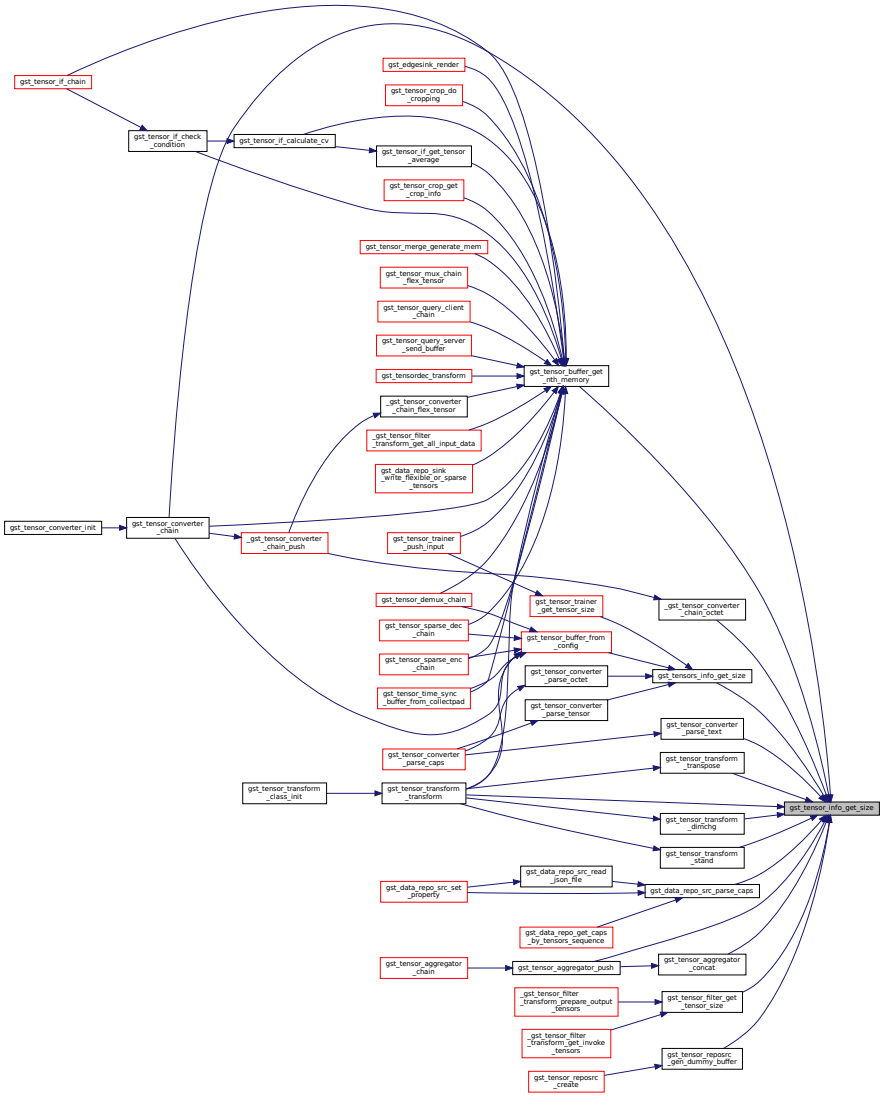
data size

Definition at line 156 of file nnstreamer_plugin_api_util_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.94.3.23 gst_tensor_info_init()

```
void gst_tensor_info_init (
            GstTensorInfo * info )
```

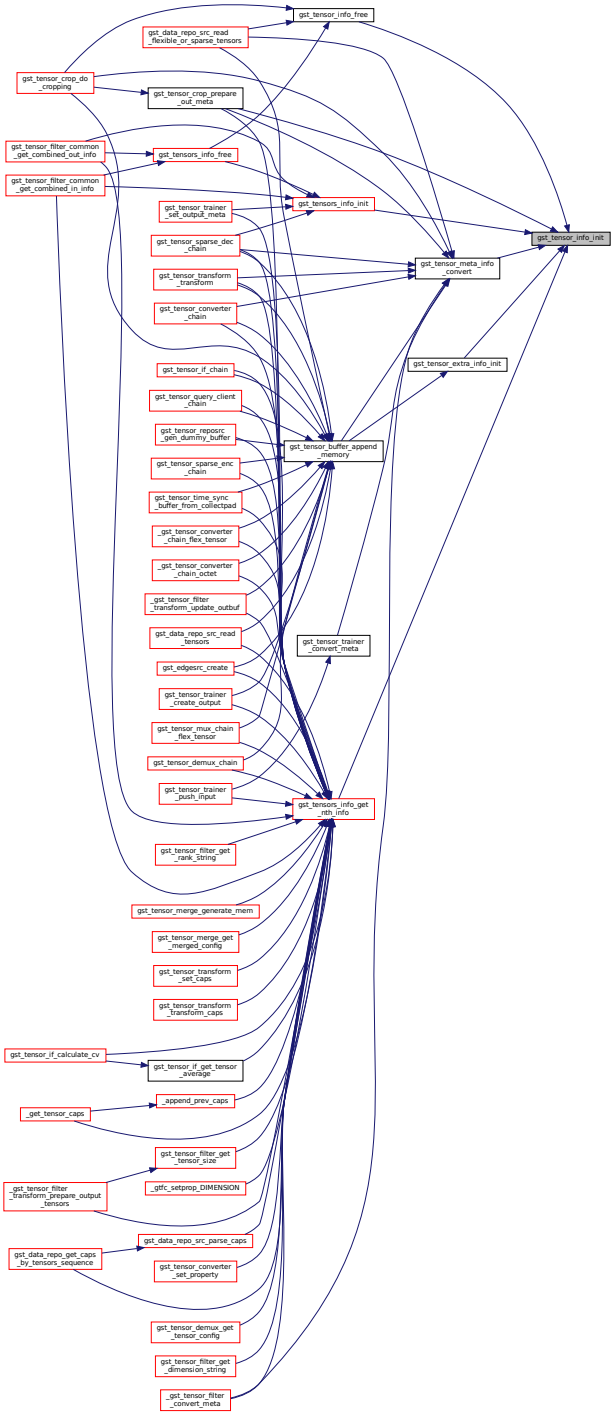
Initialize the tensor info structure.

Parameters

info	tensor info structure to be initialized

Definition at line 121 of file nnstreamer_plugin_api_util_impl.c.

Here is the caller graph for this function:



9.94.3.24 gst_tensor_info_is_equal()

```
gboolean gst_tensor_info_is_equal (
            const GstTensorInfo * i1,
            const GstTensorInfo * i2 )
```

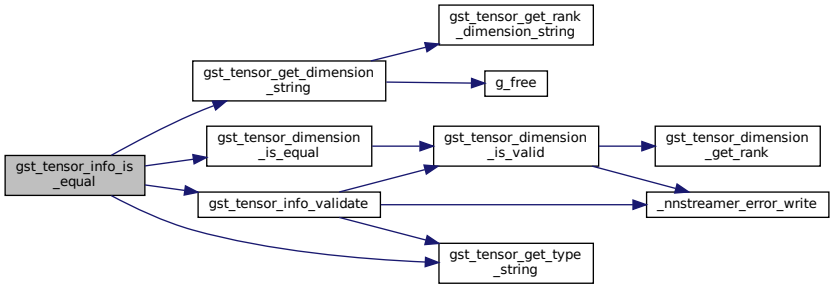
Compare tensor info.

Returns

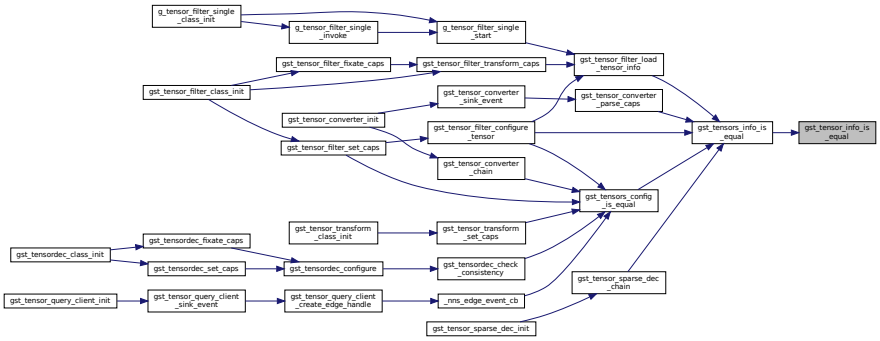
TRUE if equal, FALSE if given tensor infos are invalid or not equal.

Definition at line 197 of file nnstreamer_plugin_api_util_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.94.3.25 gst_tensor_info_validate()

```
gboolean gst_tensor_info_validate (
            const GstTensorInfo * info )
```

Check the tensor info is valid.

Parameters

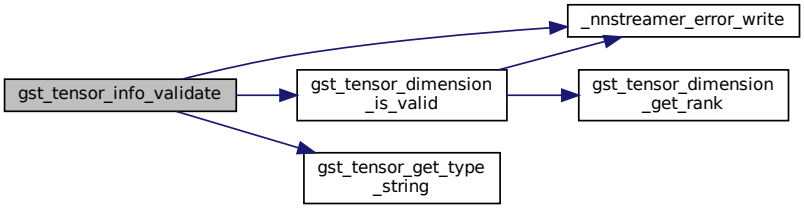
info	tensor info structure

Returns

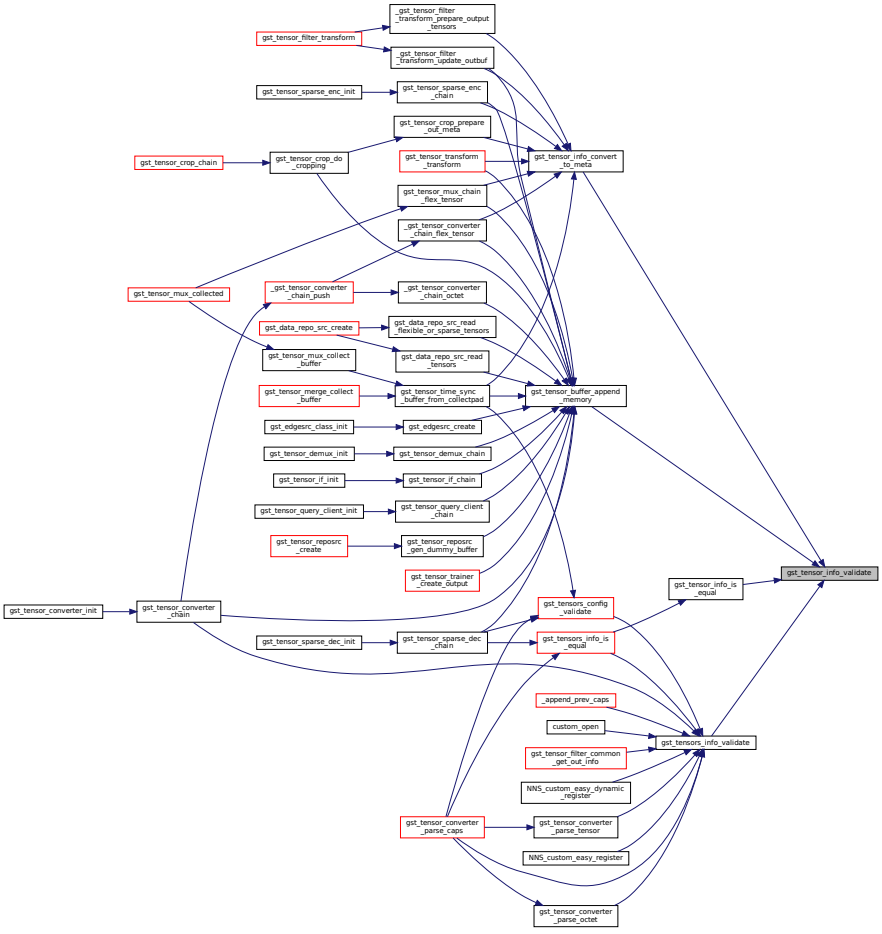
TRUE if info is valid

Definition at line 174 of file nnstreamer_plugin_api_util_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.94.3.26 gst_tensor_meta_info_convert()

```
gboolean gst_tensor_meta_info_convert (
            GstTensorMetaInfo * meta,
            GstTensorInfo * info )
```

Convert GstTensorMetaInfo structure to GstTensorInfo.

Parameters

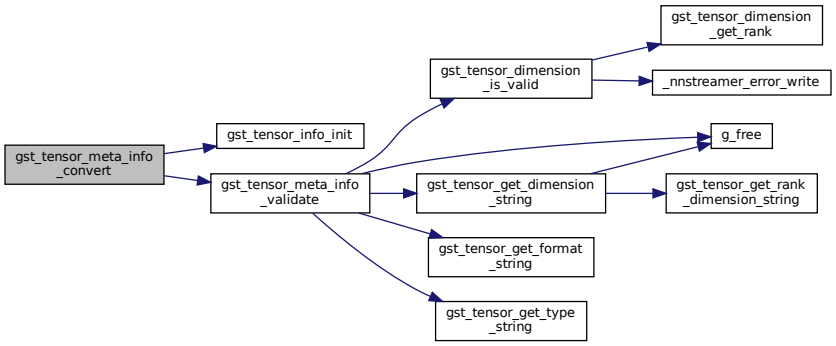
in	*meta*	tensor meta structure to be converted
out	*info*	GstTensorInfo to be filled

Returns

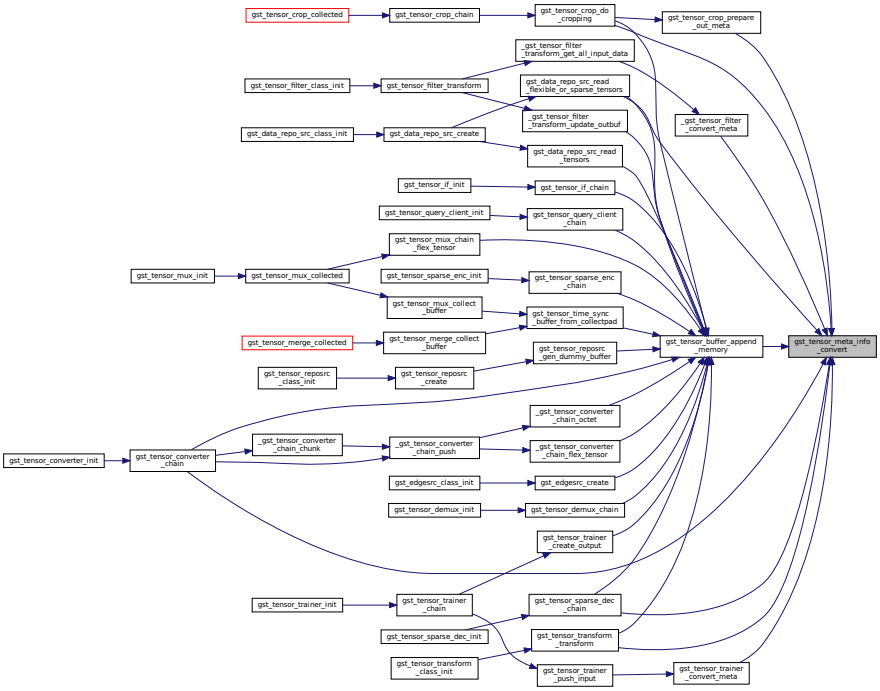
TRUE if successfully set the info

Definition at line 1562 of file nnstreamer_plugin_api_util_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.94.3.27 gst_tensor_meta_info_get_data_size()

```
gsize gst_tensor_meta_info_get_data_size (
            GstTensorMetaInfo * meta )
```

Get the data size calculated from tensor meta.

Parameters

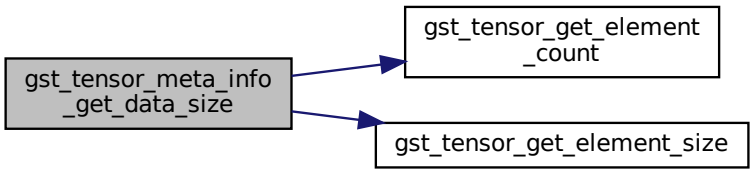
in	*meta*	tensor meta structure

Returns

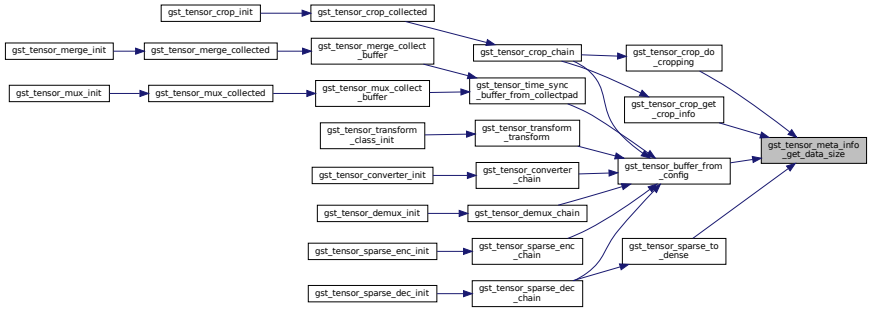
The data size for meta info (0 if meta is invalid)

Definition at line 1477 of file nnstreamer_plugin_api_util_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.94.3.28 gst_tensor_meta_info_get_header_size()

```
gsize gst_tensor_meta_info_get_header_size (
            GstTensorMetaInfo * meta )
```

Get the header size to handle a tensor meta.

Parameters

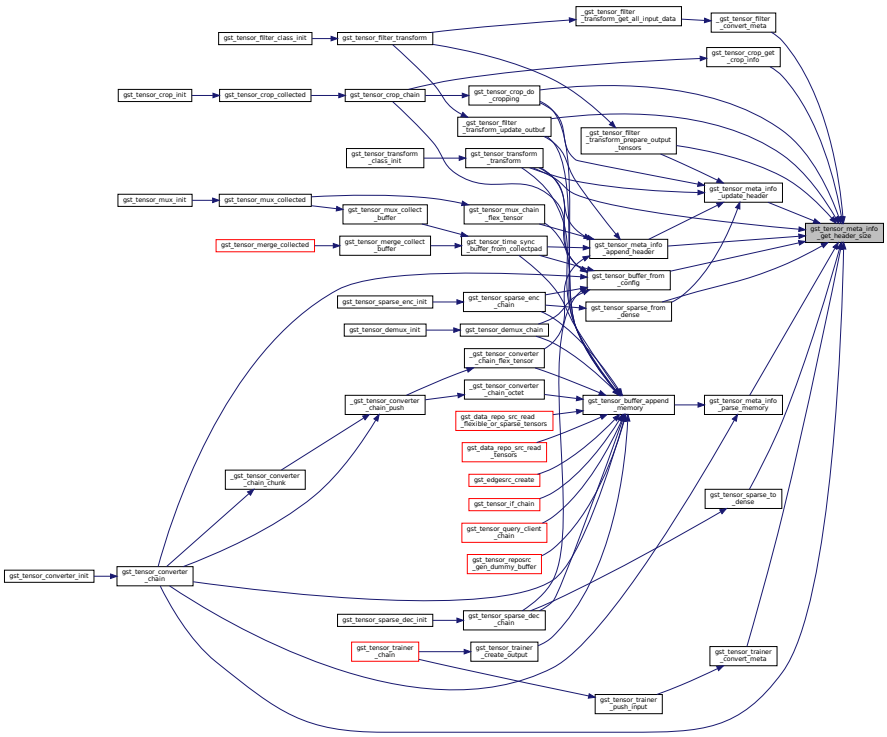
in	*meta*	tensor meta structure

Returns

Header size for meta info (0 if meta is invalid)

Definition at line 1456 of file nnstreamer_plugin_api_util_impl.c.

Here is the caller graph for this function:



9.94.3.29 gst_tensor_meta_info_get_version()

```
void gst_tensor_meta_info_get_version (
            GstTensorMetaInfo * meta,
            guint * major,
            guint * minor )
```

Get the version of tensor meta.

Parameters

in	*meta*	tensor meta structure
out	*major*	pointer to get the major version number
out	*minor*	pointer to get the minor version number

Definition at line 1393 of file nnstreamer_plugin_api_util_impl.c.

9.94.3.30 gst_tensor_meta_info_init()

```
void gst_tensor_meta_info_init (
```

```
            GstTensorMetaInfo * meta )
```

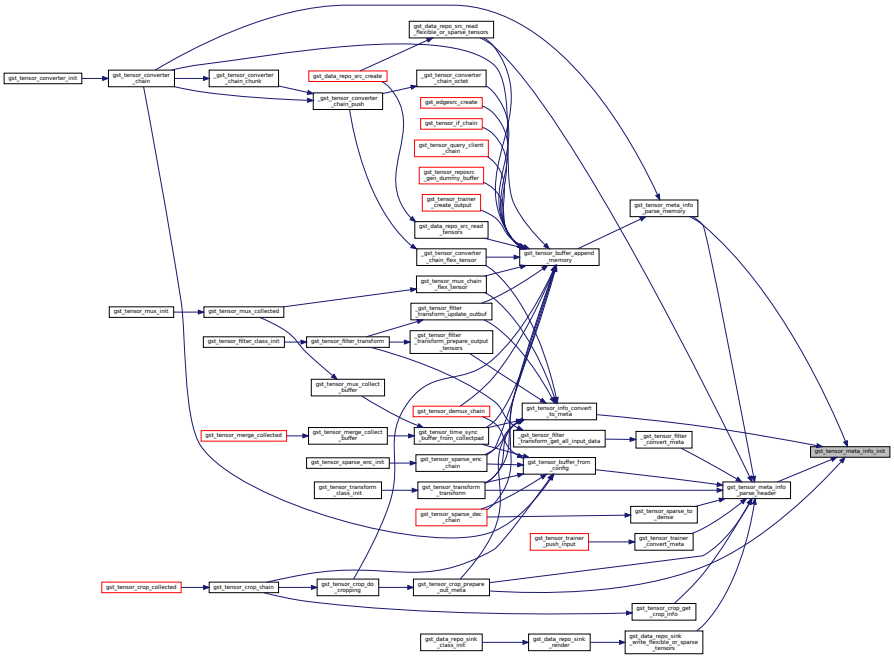
Initialize the tensor meta info structure.

Parameters

in,out	*meta*	tensor meta structure to be initialized

Definition at line 1372 of file nnstreamer_plugin_api_util_impl.c.

Here is the caller graph for this function:



9.94.3.31 gst_tensor_meta_info_parse_header()

```
gboolean gst_tensor_meta_info_parse_header (
            GstTensorMetaInfo * meta,
            gpointer header )
```

Parse header and fill the tensor meta.

Parameters

out	*meta*	tensor meta structure to be filled
in	*header*	pointer to header to be parsed

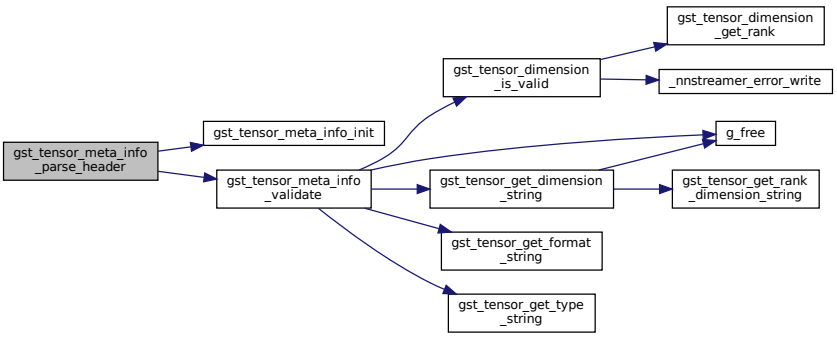
Returns

TRUE if successfully set the meta

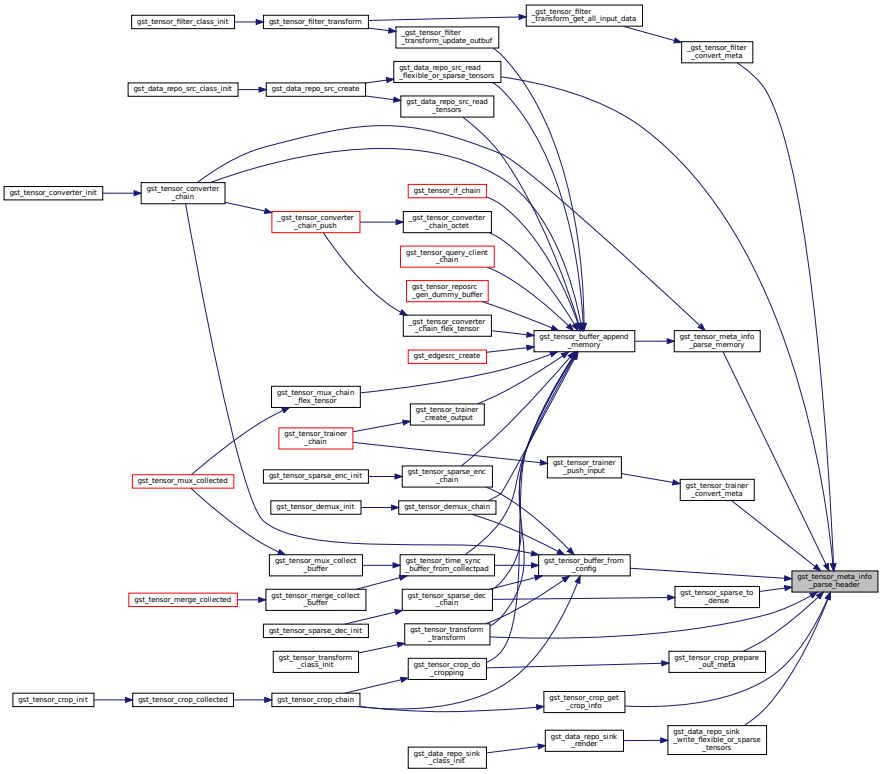
Todo update meta info for each version

Definition at line 1527 of file nnstreamer_plugin_api_util_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.94.3.32 gst_tensor_meta_info_update_header()

```
gboolean gst_tensor_meta_info_update_header (
            GstTensorMetaInfo * meta,
            gpointer header )
```

Update header from tensor meta.

Parameters

in	*meta*	tensor meta structure
out	*header*	pointer to header to be updated

Returns

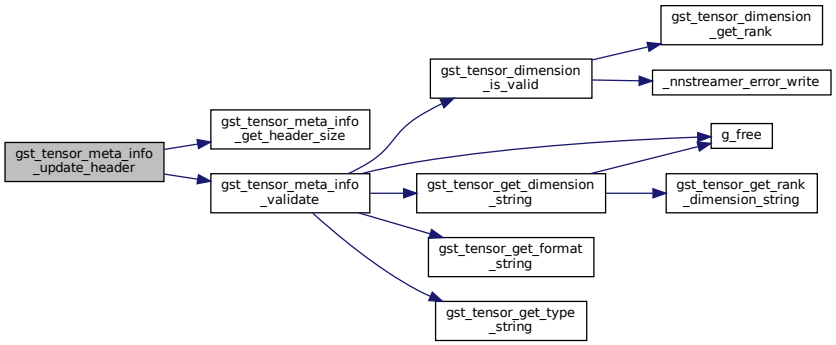
TRUE if successfully set the header

Note

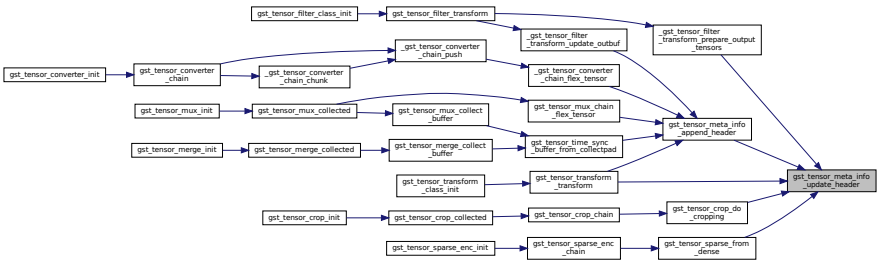
User should allocate enough memory for header (see gst_tensor_meta_info_get_header_size()).

Definition at line 1505 of file nnstreamer_plugin_api_util_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.94.3.33 gst_tensor_meta_info_validate()

```
gboolean gst_tensor_meta_info_validate (
            GstTensorMetaInfo * meta )
```

Check the meta info is valid.

Parameters

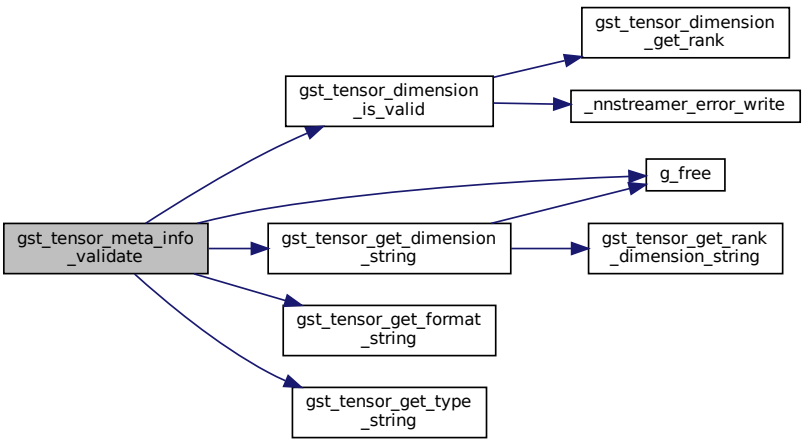
in	*meta*	tensor meta structure

Returns

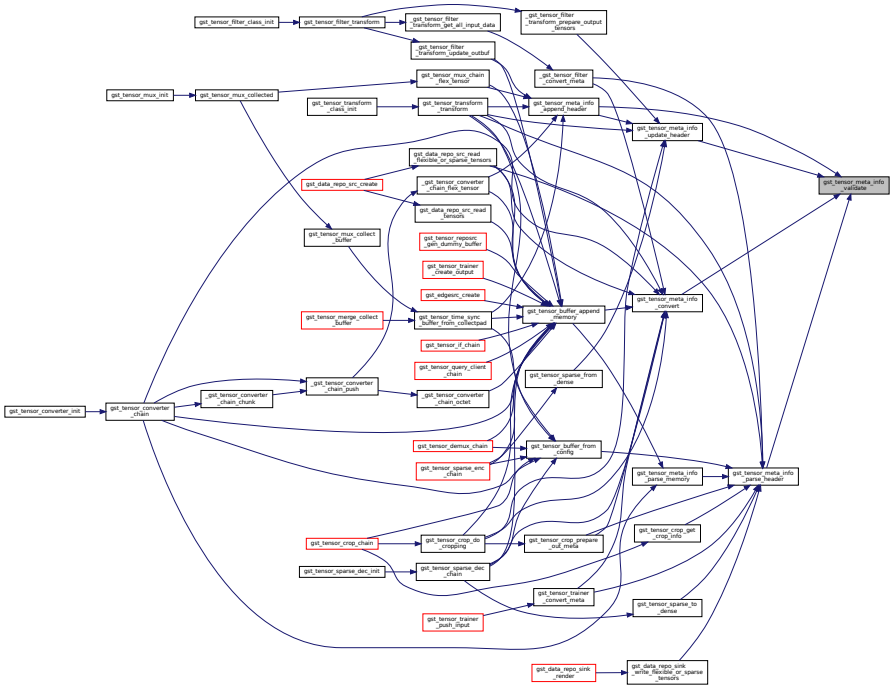
TRUE if given meta is valid

Definition at line 1414 of file nnstreamer_plugin_api_util_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.94.3.34 gst_tensor_parse_dimension()

```
guint gst_tensor_parse_dimension (
            const gchar * dimstr,
            tensor_dim dim )
```

Parse tensor dimension parameter string.

Returns

The Rank. 0 if error.

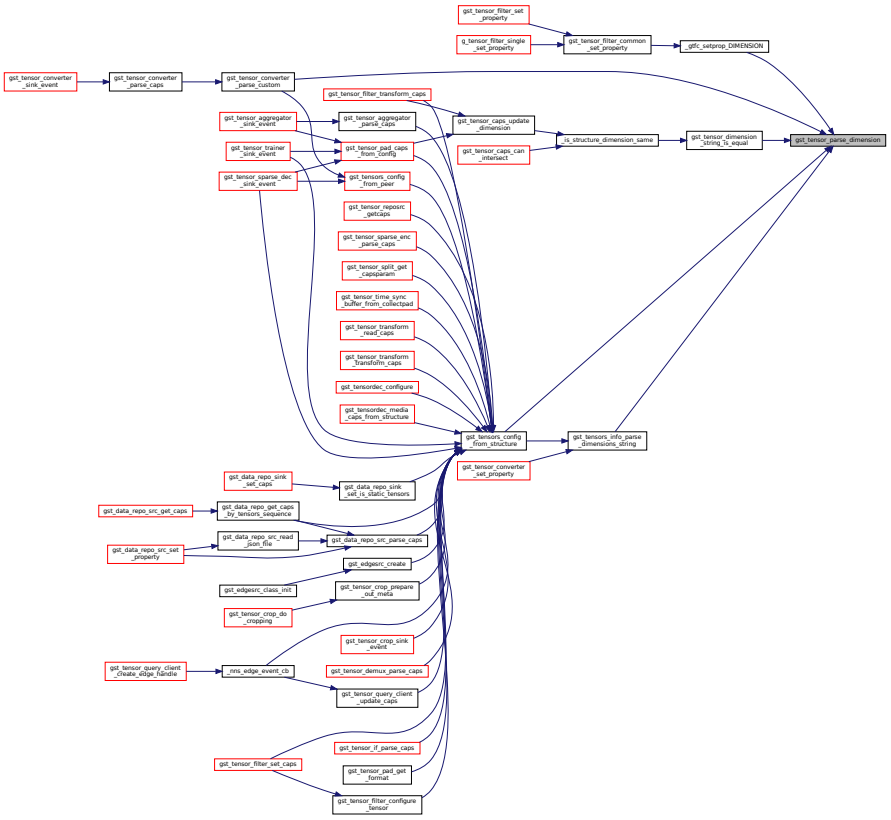
Parameters

dimstr	The dimension string in the format of d1:...:d16, d1:d2:d3, d1:d2, or d1, where dN is a positive integer and d1 is the innermost dimension; i.e., dim[d16]...[d1];
dim	dimension to be filled.

Definition at line 1040 of file nnstreamer_plugin_api_util_impl.c.

Here is the call graph for this function:

Here is the caller graph for this function:



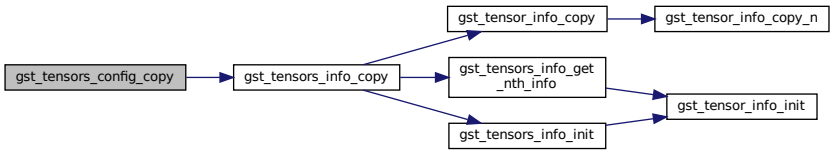
9.94.3.35 gst_tensors_config_copy()

```
void gst_tensors_config_copy (
            GstTensorsConfig * dest,
            const GstTensorsConfig * src )
```

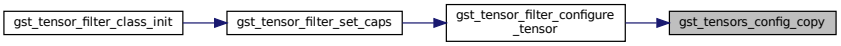
Copy tensors config.

Definition at line 904 of file nnstreamer_plugin_api_util_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.94.3.36 gst_tensors_config_free()

```
void gst_tensors_config_free (
            GstTensorsConfig * config )
```

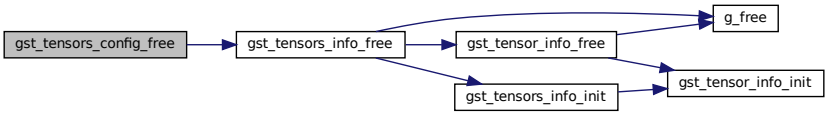
Free allocated data in tensors config structure.

Parameters

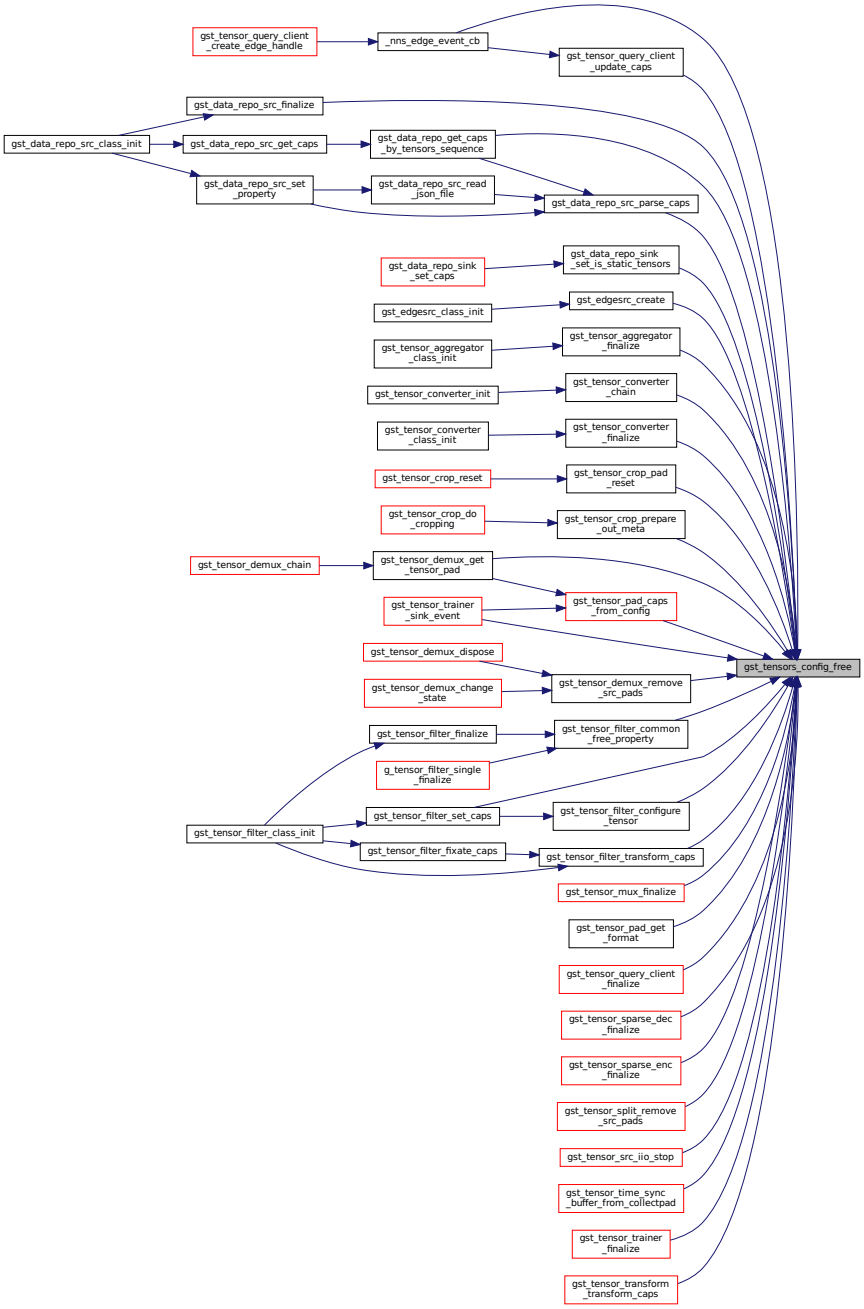
config	tensors config structure

Definition at line 845 of file nnstreamer_plugin_api_util_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.94.3.37 gst_tensors_config_init()

```
void gst_tensors_config_init (
            GstTensorsConfig * config )
```

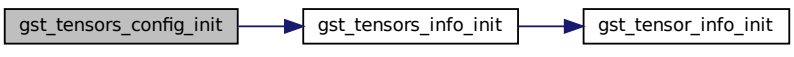
Initialize the tensors config info structure (for other/tensors)

Parameters

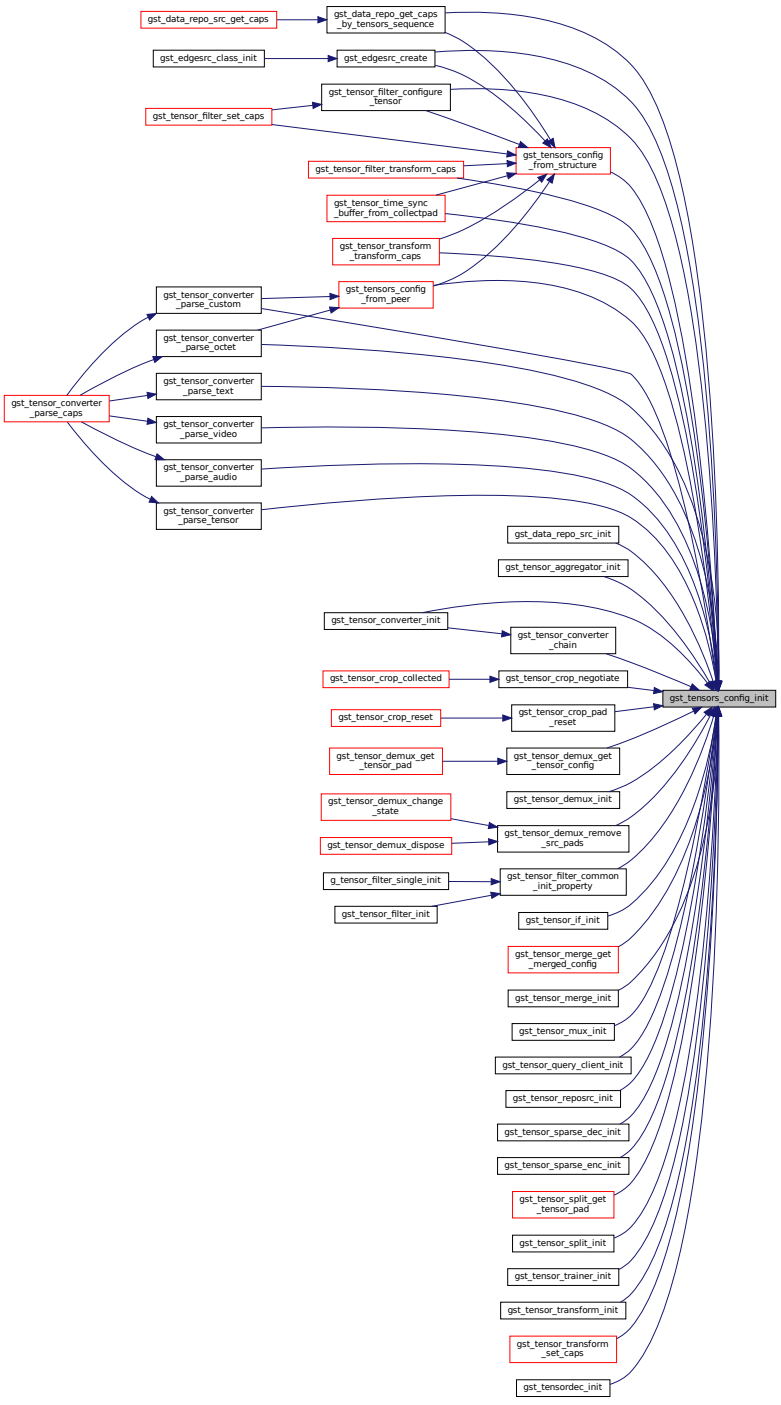
config	tensors config structure to be initialized

Definition at line 830 of file nnstreamer_plugin_api_util_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.94.3.38 gst_tensors_config_is_equal()

```
gboolean gst_tensors_config_is_equal (
            const GstTensorsConfig * c1,
            const GstTensorsConfig * c2 )
```

Compare tensor config info.

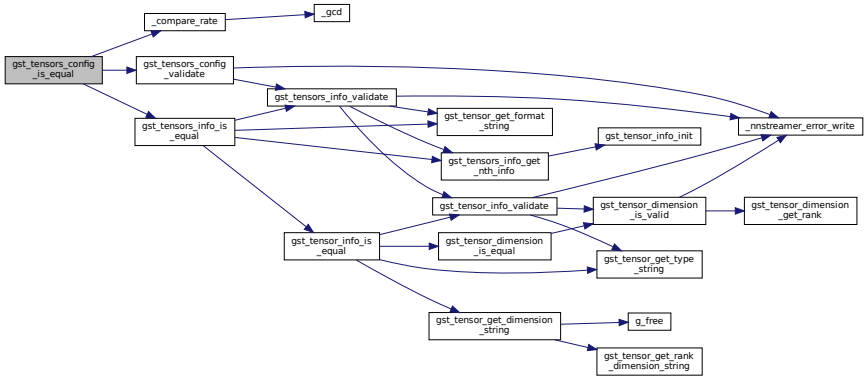
Compare tensor config info (for other/tensors)

Parameters

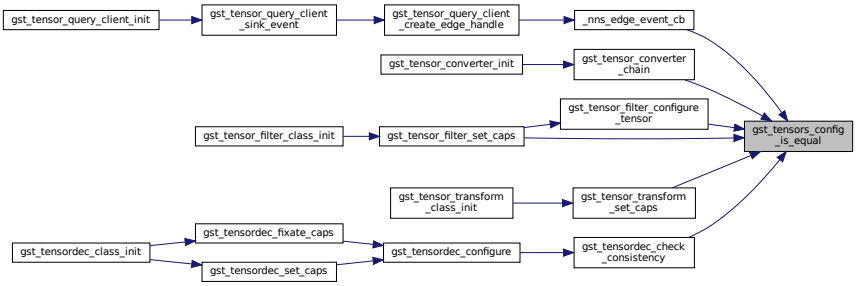
TRUE	if equal

Definition at line 881 of file nnstreamer_plugin_api_util_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.94.3.39 gst_tensors_config_to_string()

```
gchar* gst_tensors_config_to_string (
            const GstTensorsConfig * config )
```

Tensor config represented as a string. Caller should free it.

Parameters

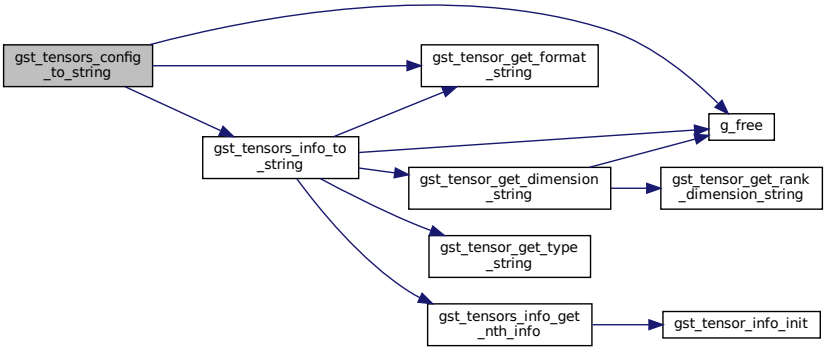
config	tensor config structure

Returns

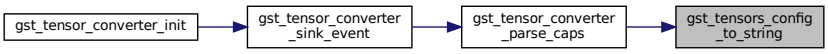
The newly allocated string representing the config. Free after use.

Definition at line 920 of file nnstreamer_plugin_api_util_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.94.3.40 gst_tensors_config_validate()

```
gboolean gst_tensors_config_validate (
            const GstTensorsConfig * config )
```

Check the tensors are all configured.

Check the tensors are all configured (for other/tensors)

Parameters

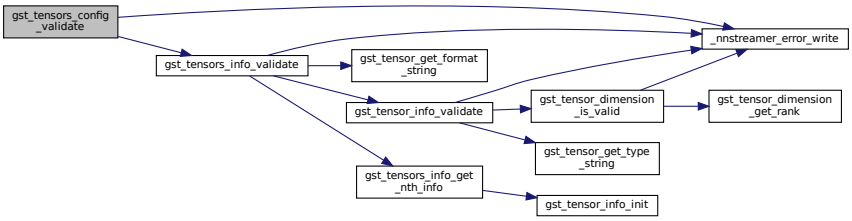
config	tensor config structure

Returns

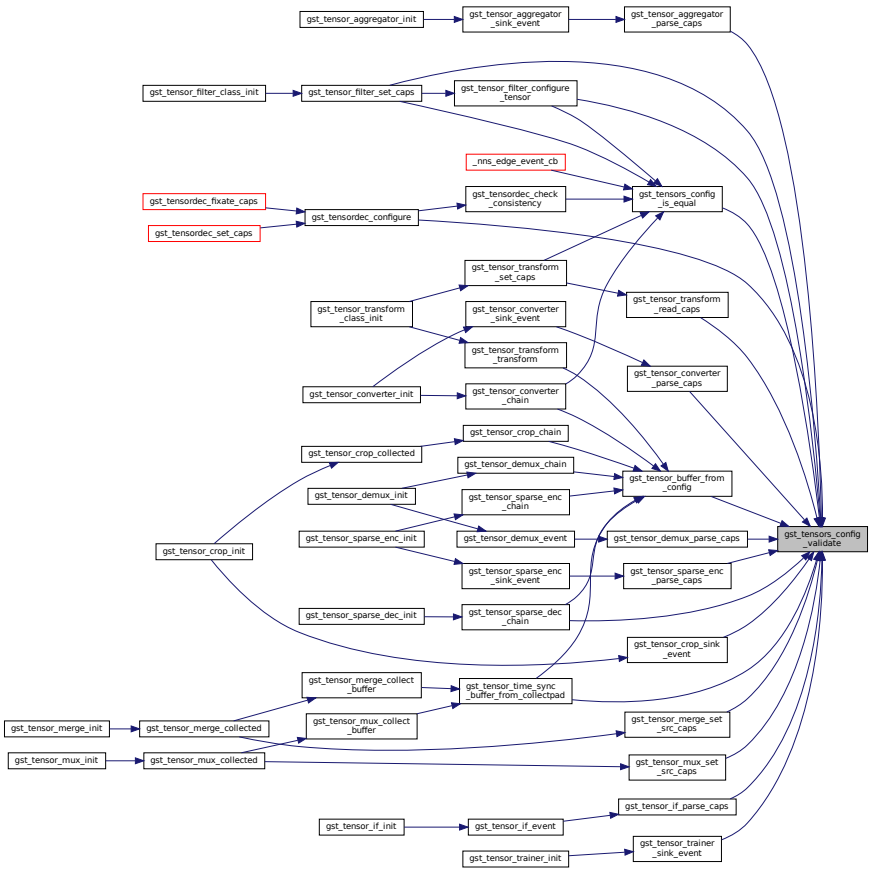
TRUE if configured

Definition at line 858 of file nnstreamer_plugin_api_util_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.94.3.41 gst_tensors_info_copy()

```
void gst_tensors_info_copy (
            GstTensorsInfo * dest,
            const GstTensorsInfo * src )
```

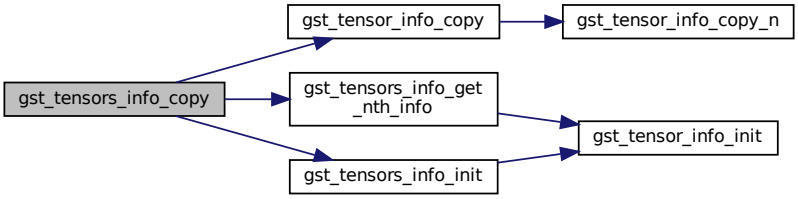
Copy tensor info.

Note

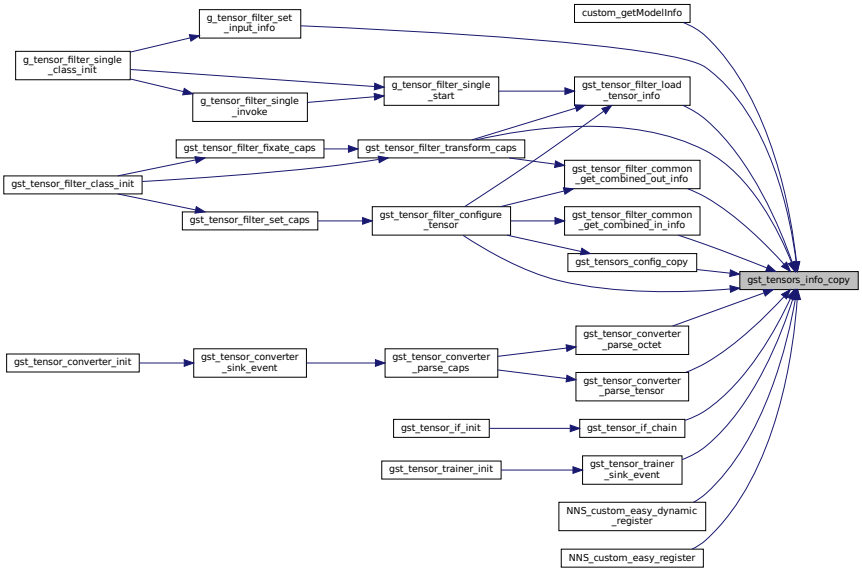
Copied info should be freed with gst_tensors_info_free()

Definition at line 502 of file nnstreamer_plugin_api_util_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.94.3.42 gst_tensors_info_free()

```
void gst_tensors_info_free (
            GstTensorsInfo * info )
```

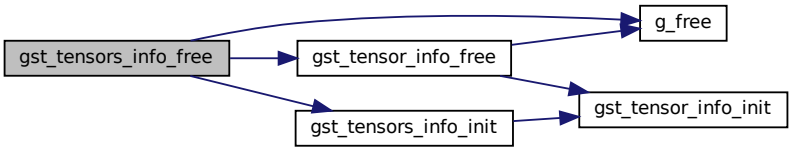
Free allocated data in tensors info structure.

Parameters

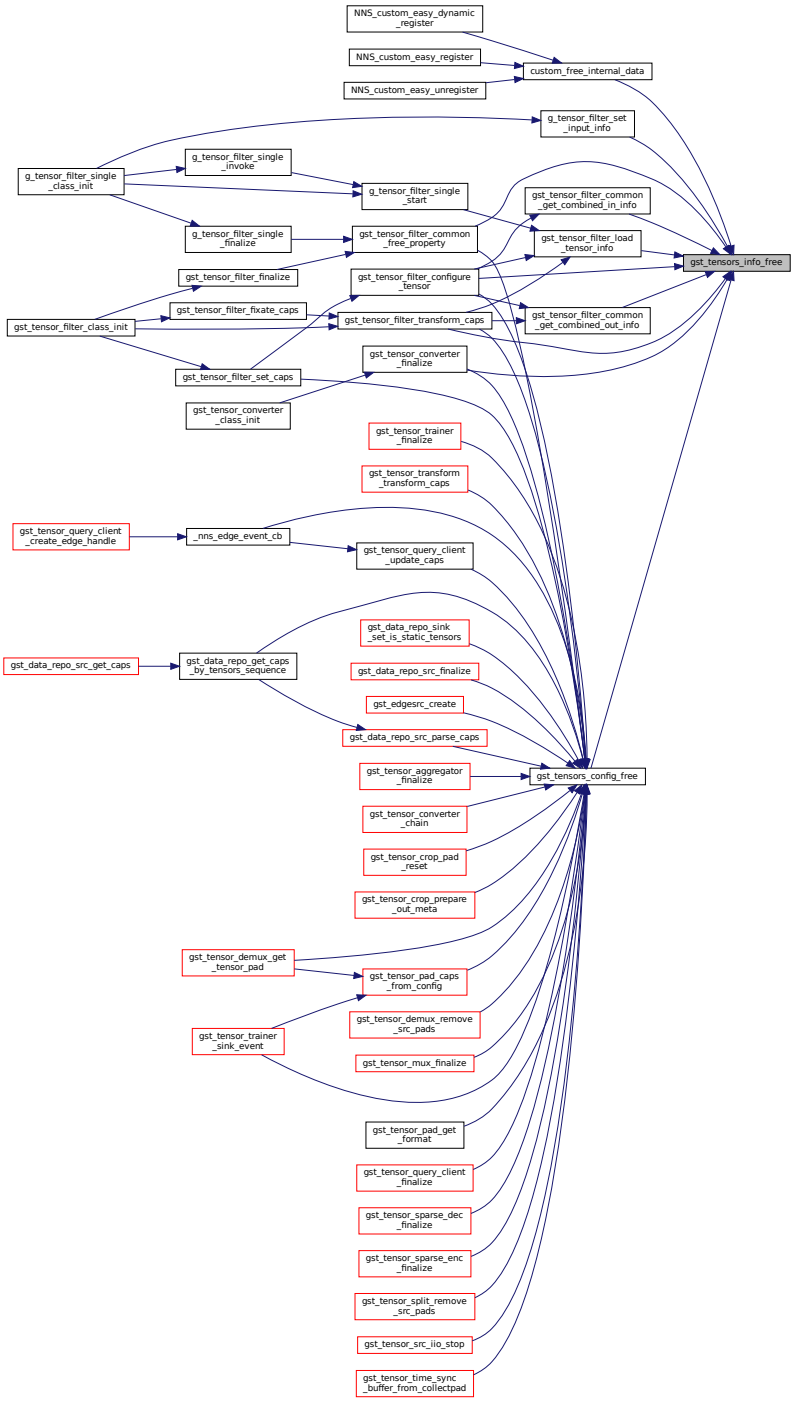
info	tensors info structure

Definition at line 347 of file nnstreamer_plugin_api_util_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.94.3.43 gst_tensors_info_get_dimensions_string()

```
gchar* gst_tensors_info_get_dimensions_string (
            const GstTensorsInfo * info )
```

Get the string of dimensions in tensors info.

Parameters

info	tensors info structure

Returns

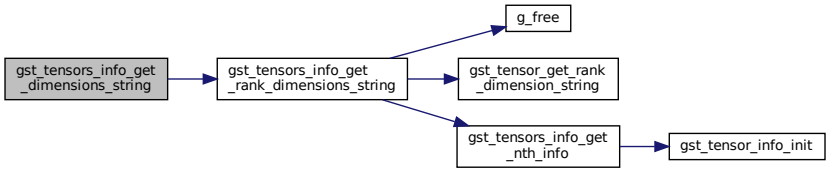
string of dimensions in tensors info (NULL if the number of tensors is 0)

Note

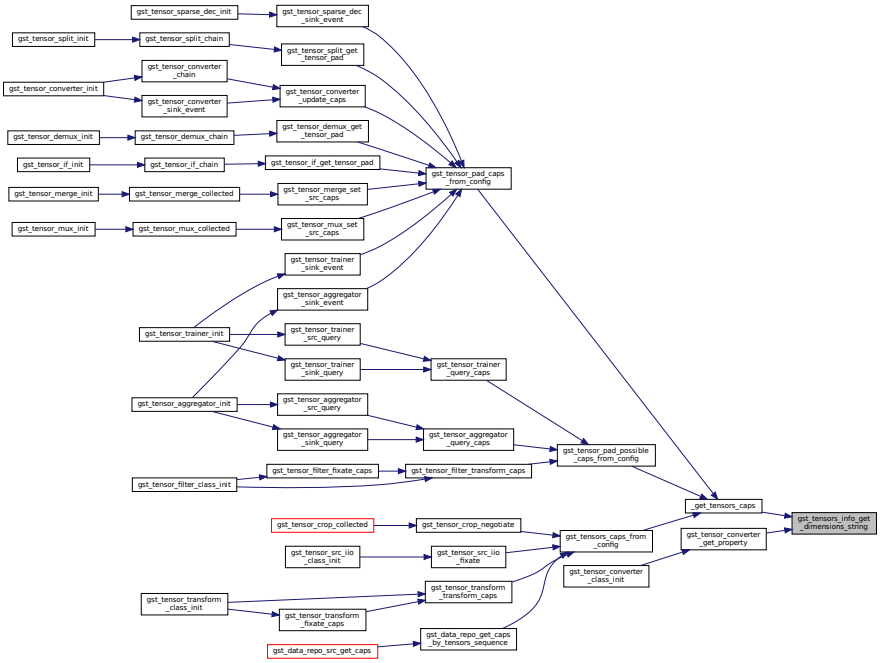
The returned value should be freed with g_free()

Definition at line 661 of file nnstreamer_plugin_api_util_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.94.3.44 gst_tensors_info_get_names_string()

```
gchar* gst_tensors_info_get_names_string (
            const GstTensorsInfo * info )
```

Get the string of tensor names in tensors info.

Parameters

info	tensors info structure

Returns

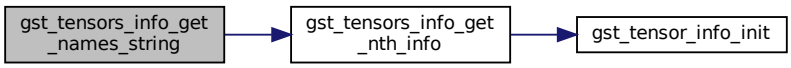
string of names in tensors info (NULL if the number of tensors is 0)

Note

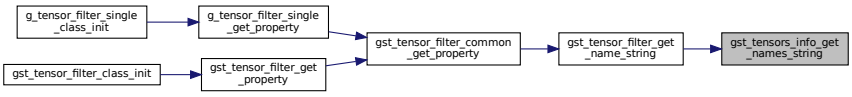
The returned value should be freed with g_free()

Definition at line 749 of file nnstreamer_plugin_api_util_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.94.3.45 gst_tensors_info_get_nth_info()

```
GstTensorInfo* gst_tensors_info_get_nth_info (
            GstTensorsInfo * info,
            guint index )
```

Get the pointer of nth tensor information.

Parameters

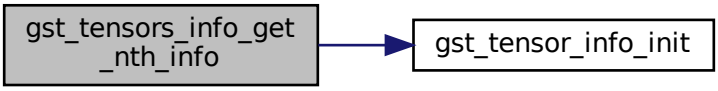
info	tensors info structure
index	the index of tensor to be fetched

Returns

The pointer to tensor info structure

Definition at line 296 of file nnstreamer_plugin_api_util_impl.c.

Here is the call graph for this function:



9.94.3.46 gst_tensors_info_get_rank_dimensions_string()

```
gchar* gst_tensors_info_get_rank_dimensions_string (
            const GstTensorsInfo * info,
            const unsigned int rank )
```

Get the string of dimensions in tensors info and rank count.

Parameters

info	tensors info structure
rank	rank count of given tensor dimension

Returns

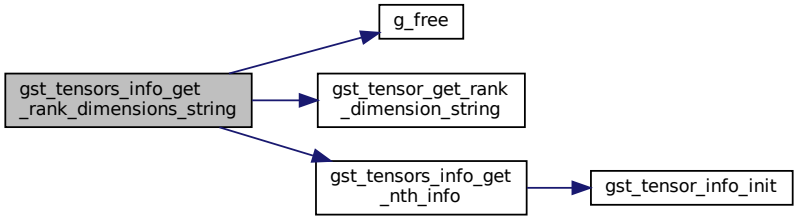
Formatted string of given dimension

Note

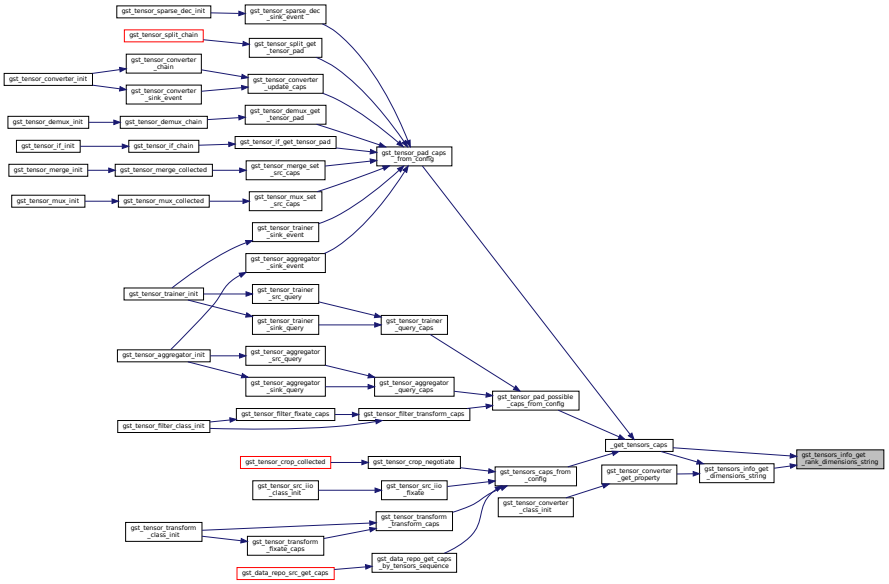
If rank count is 3, then returned string is 'd1:d2:d3`. The returned value should be freed with g_free()

Definition at line 676 of file nnstreamer_plugin_api_util_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.94.3.47 gst_tensors_info_get_size()

```
gsize gst_tensors_info_get_size (
            const GstTensorsInfo * info,
            gint index )
```

Get data size of single tensor.

Parameters

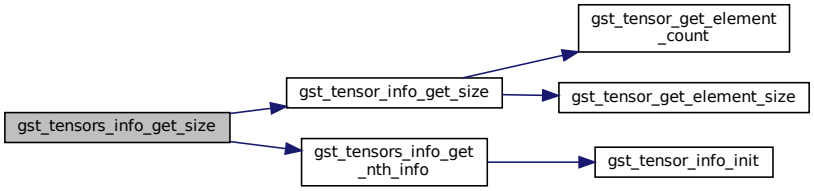
info	tensors info structure
index	the index of tensor (-1 to get total size of tensors)

Returns

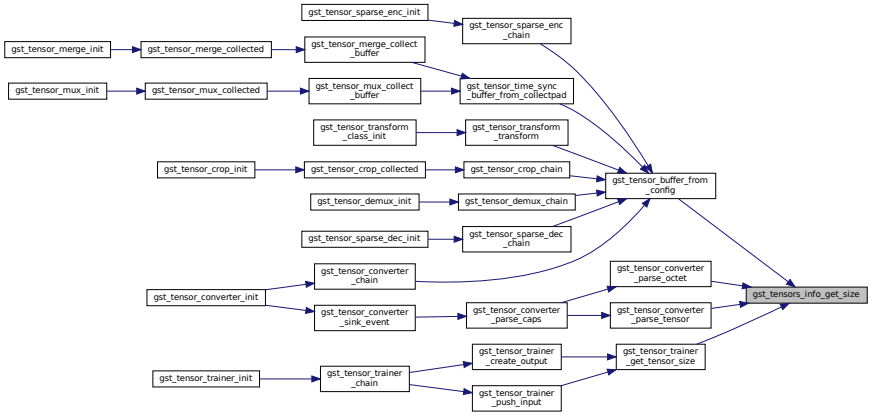
data size

Definition at line 376 of file nnstreamer_plugin_api_util_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.94.3.48 gst_tensors_info_get_types_string()

```
gchar* gst_tensors_info_get_types_string (
            const GstTensorsInfo * info )
```

Get the string of types in tensors info.

Parameters

info	tensors info structure

Returns

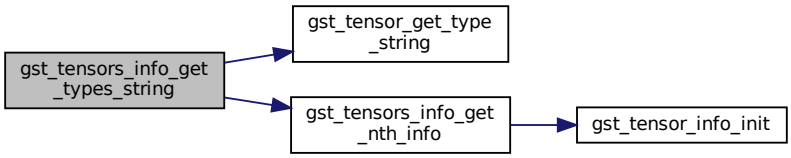
string of types in tensors info (NULL if the number of tensors is 0)

Note

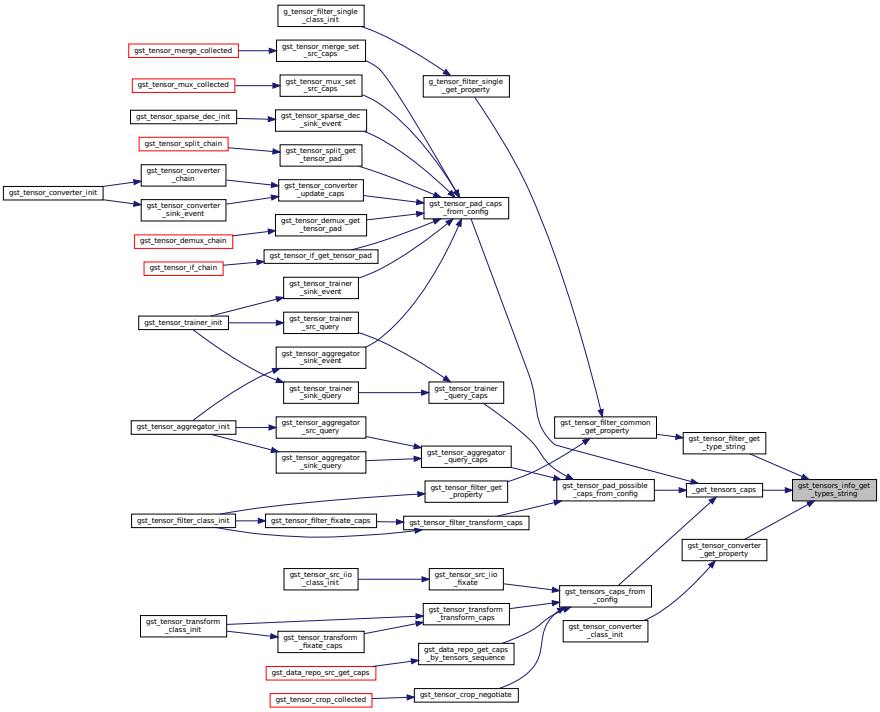
The returned value should be freed with g_free()

Definition at line 714 of file nnstreamer_plugin_api_util_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.94.3.49 gst_tensors_info_init()

```
void gst_tensors_info_init (
            GstTensorsInfo * info )
```

Initialize the tensors info structure.

Parameters

info	tensors info structure to be initialized

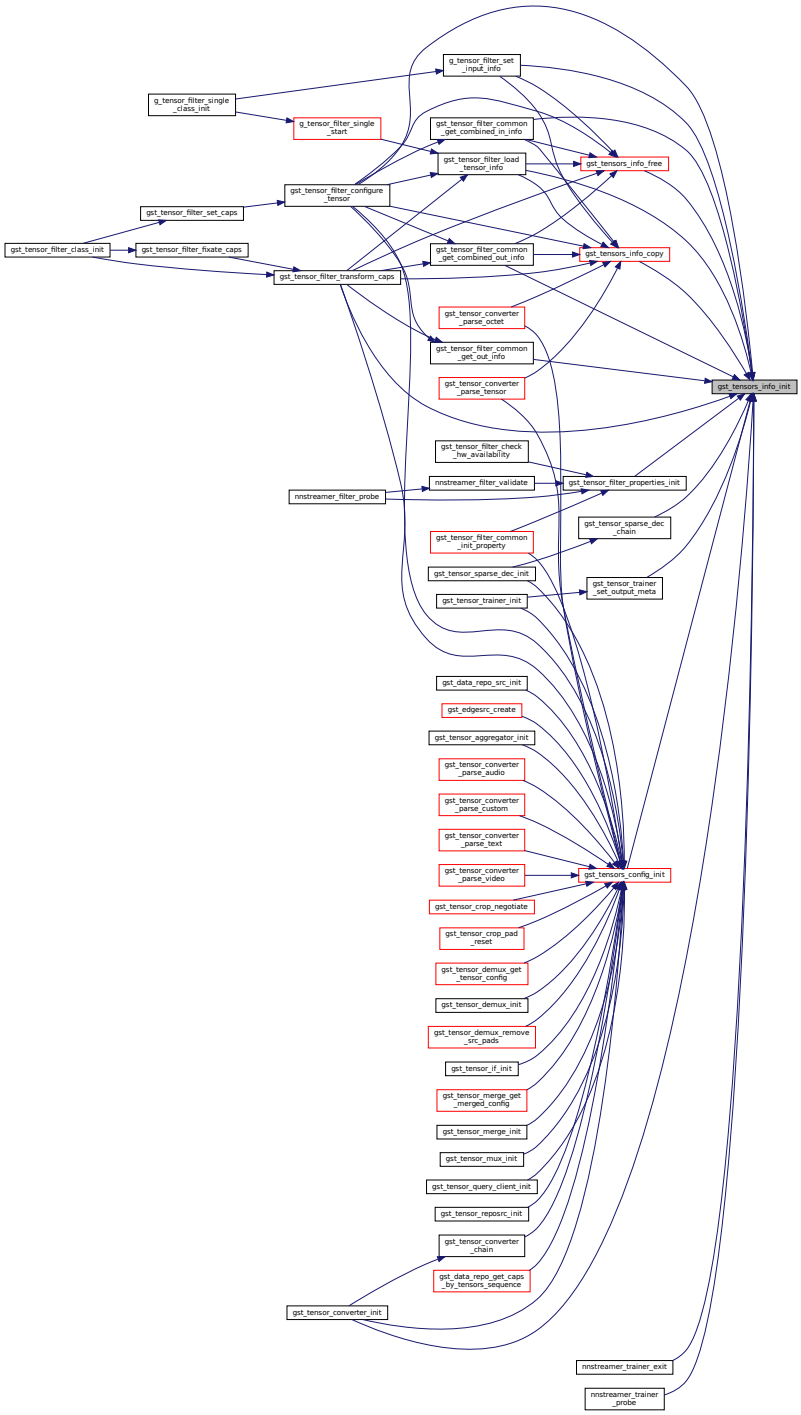
Note

default format is static

Definition at line 325 of file nnstreamer_plugin_api_util_impl.c.

Here is the call graph for this function:

Here is the caller graph for this function:



9.94.3.50 gst_tensors_info_is_equal()

```
gboolean gst_tensors_info_is_equal (
            const GstTensorsInfo * i1,
            const GstTensorsInfo * i2 )
```

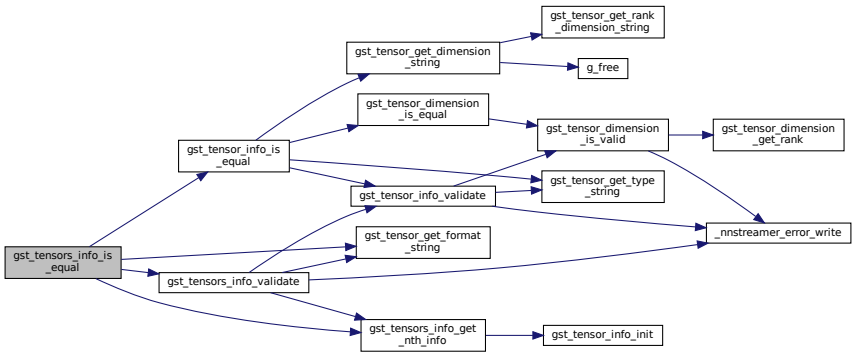
Compare tensors info.

Returns

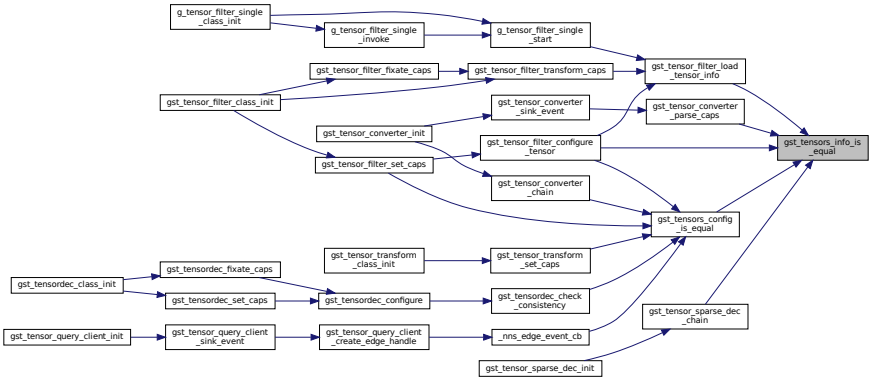
TRUE if equal, FALSE if given tensor infos are invalid or not equal.

Definition at line 454 of file nnstreamer_plugin_api_util_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.94.3.51 gst_tensors_info_parse_dimensions_string()

```
guint gst_tensors_info_parse_dimensions_string (
            GstTensorsInfo * info,
            const gchar * dim_string )
```

Parse the string of dimensions.

Parameters

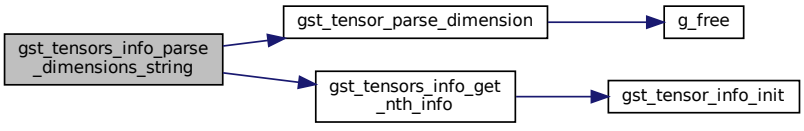
info	tensors info structure
dim_string	string of dimensions

Returns

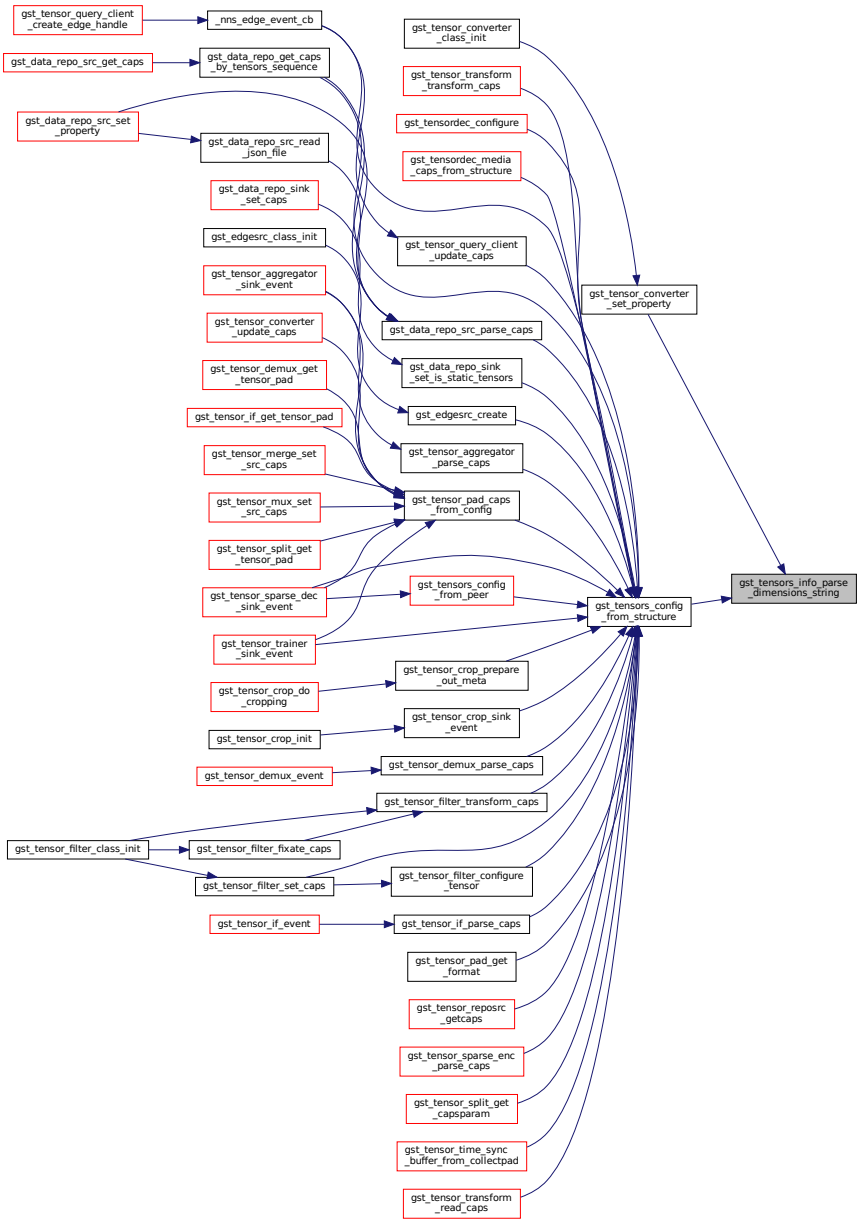
number of parsed dimensions

Definition at line 532 of file nnstreamer_plugin_api_util_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.94.3.52 gst_tensors_info_parse_names_string()

```
guint gst_tensors_info_parse_names_string (
            GstTensorsInfo * info,
            const gchar * name_string )
```

Parse the string of names.

Parameters

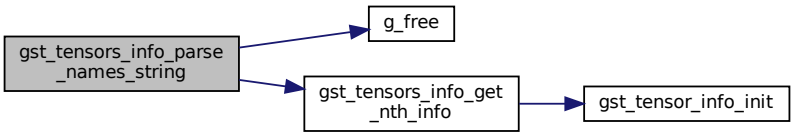
info	tensors info structure
name_string	string of names

Returns

number of parsed names

Definition at line 612 of file nnstreamer_plugin_api_util_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:

9.94.3.53 gst_tensors_info_parse_types_string()

```
guint gst_tensors_info_parse_types_string (
            GstTensorsInfo * info,
            const gchar * type_string )
```

Parse the string of types.

Parameters

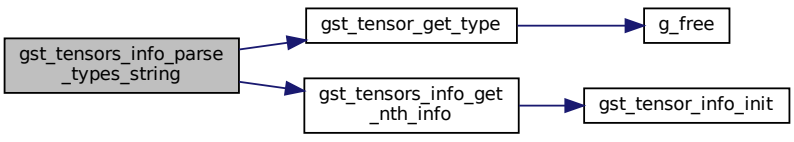
info	tensors info structure
type_string	string of types

Returns

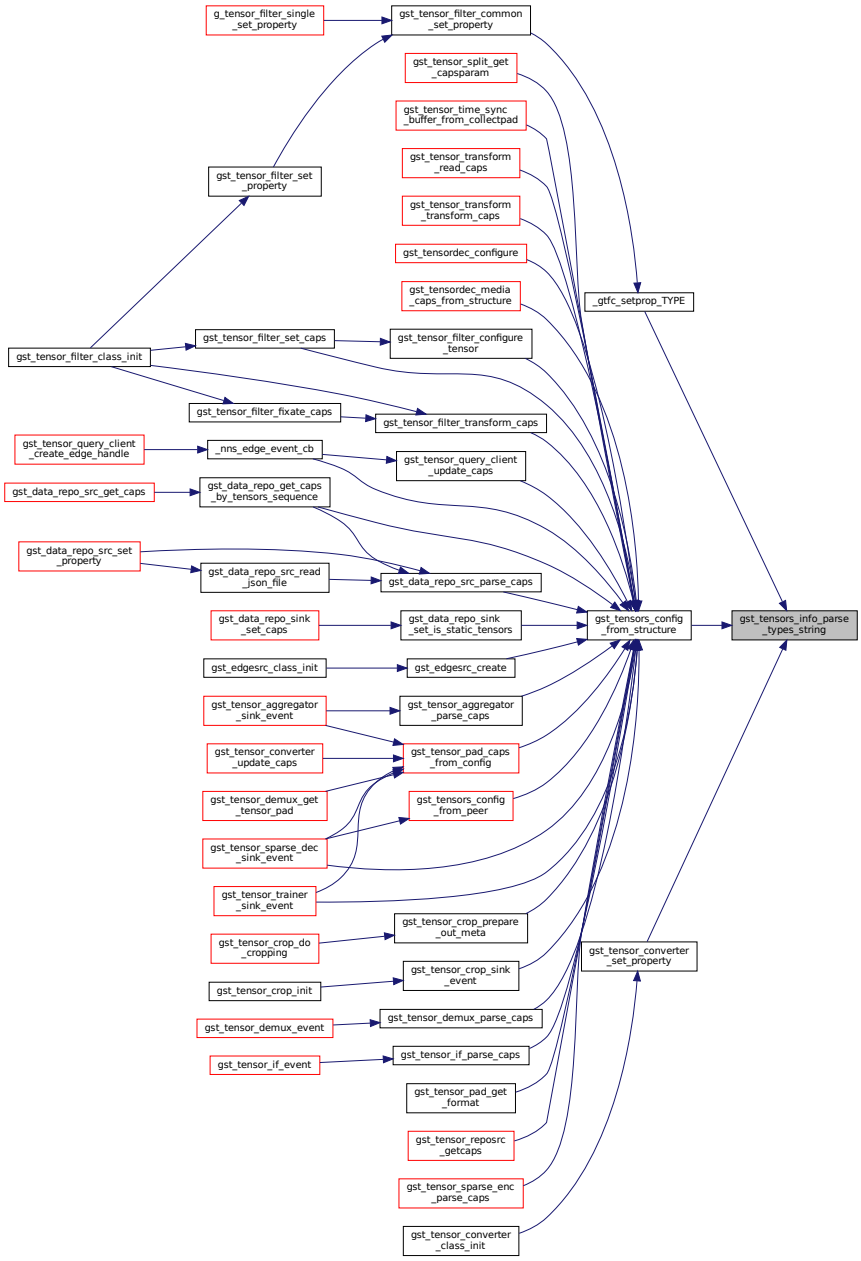
number of parsed types

Definition at line 572 of file nnstreamer_plugin_api_util_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.94.3.54 gst_tensors_info_to_string()

```
gchar* gst_tensors_info_to_string (
            const GstTensorsInfo * info )
```

GstTensorsInfo represented as a string. Caller should free it.

Parameters

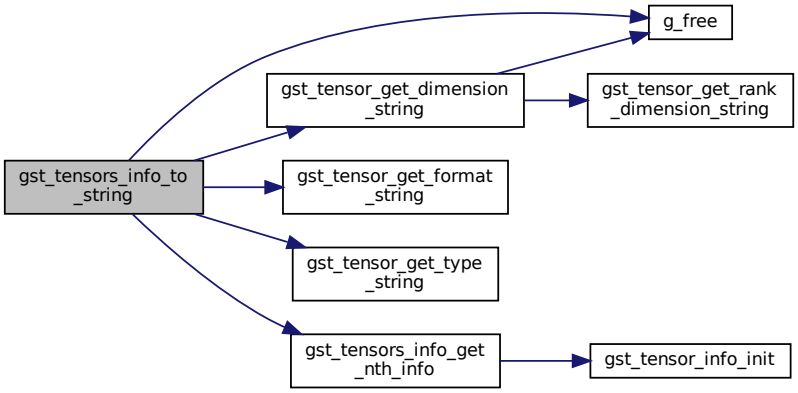
info	GstTensorsInfo structure

Returns

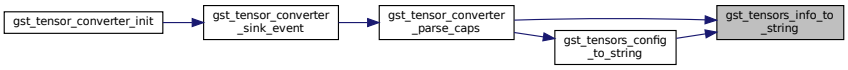
The newly allocated string representing the tensors info. Free after use.

Definition at line 783 of file nnstreamer_plugin_api_util_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.94.3.55 gst_tensors_info_validate()

```
gboolean gst_tensors_info_validate (
            const GstTensorsInfo * info )
```

Check the tensors info is valid.

Parameters

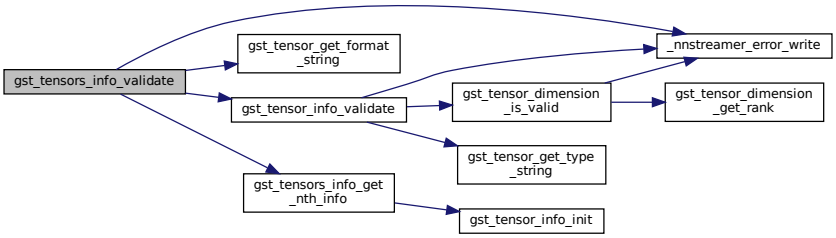
info	tensors info structure

Returns

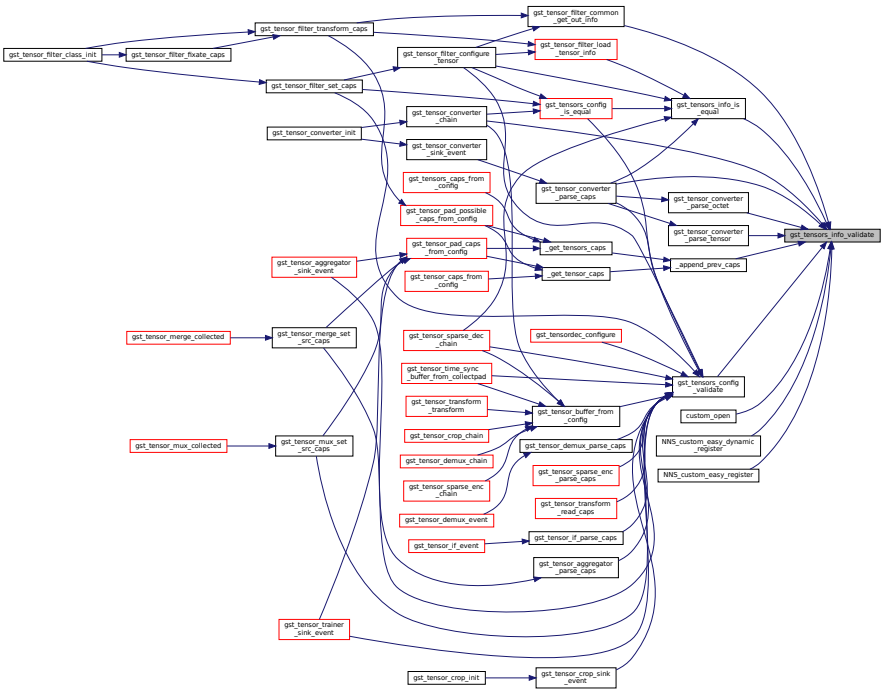
TRUE if info is valid

Definition at line 404 of file nnstreamer_plugin_api_util_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.94.3.56 nnstreamer_version_fetch()

```
void nnstreamer_version_fetch (
            guint * major,
            guint * minor,
            guint * micro )
```

Get the version of NNStreamer (int, divided).

Parameters

out	*major*	MAJOR.minor.micro, won't set if it's null.
out	*minor*	major.MINOR.micro, won't set if it's null.
out	*micro*	major.minor.MICRO, won't set if it's null.

Definition at line 1624 of file nnstreamer_plugin_api_util_impl.c.

9.94.3.57 nnstreamer_version_string()

```
gchar* nnstreamer_version_string (
            void  )
```

Get the version of NNStreamer (string).

Get the version of NNStreamer.

Returns

Newly allocated string. The returned string should be freed with g_free().

Definition at line 1609 of file nnstreamer_plugin_api_util_impl.c.

9.94.4 Variable Documentation

9.94.4.1 tensor_element_size

```
const guint tensor_element_size[]  [static]
```

Initial value:

```
= {
  [_NNS_INT32] = 4,
  [_NNS_UINT32] = 4,
  [_NNS_INT16] = 2,
  [_NNS_UINT16] = 2,
  [_NNS_INT8] = 1,
  [_NNS_UINT8] = 1,
  [_NNS_FLOAT64] = 8,
  [_NNS_FLOAT32] = 4,
  [_NNS_INT64] = 8,
  [_NNS_UINT64] = 8,
  [_NNS_FLOAT16] = 2,
  [_NNS_END] = 0,
}
```

Byte-per-element of each tensor element type.

Definition at line 38 of file nnstreamer_plugin_api_util_impl.c.

9.94.4.2 tensor_element_typename

```
const gchar* tensor_element_typename[]  [static]
```

Initial value:

```
= {
  [_NNS_INT32] = "int32",
  [_NNS_UINT32] = "uint32",
  [_NNS_INT16] = "int16",
  [_NNS_UINT16] = "uint16",
  [_NNS_INT8] = "int8",
  [_NNS_UINT8] = "uint8",
  [_NNS_FLOAT64] = "float64",
  [_NNS_FLOAT32] = "float32",
  [_NNS_INT64] = "int64",
  [_NNS_UINT64] = "uint64",
  [_NNS_FLOAT16] = "float16",
  [_NNS_END] = NULL,
}
```

String representations for each tensor element type.

Definition at line 20 of file nnstreamer_plugin_api_util_impl.c.

9.94.4.3 tensor_format_name

```
const gchar* tensor_format_name[]  [static]
```

Initial value:

```
= {
  [_NNS_TENSOR_FORMAT_STATIC] = "static",
  [_NNS_TENSOR_FORMAT_FLEXIBLE] = "flexible",
  [_NNS_TENSOR_FORMAT_SPARSE] = "sparse",
  [_NNS_TENSOR_FORMAT_END] = NULL
}
```

String representations for tensor format.

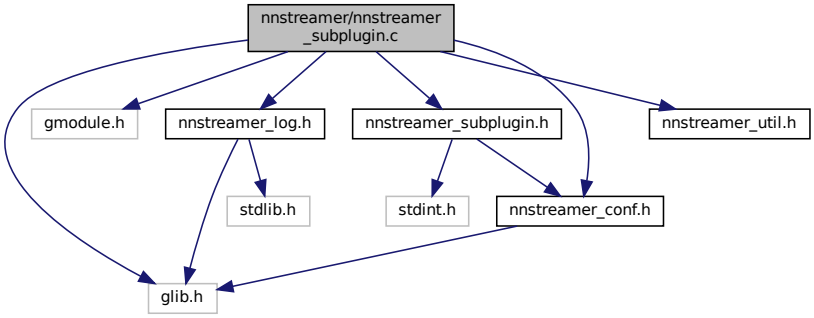
Definition at line 56 of file nnstreamer_plugin_api_util_impl.c.

9.95 nnstreamer/nnstreamer_subplugin.c File Reference

Subplugin Manager for NNStreamer.

```
#include <glib.h>
#include <gmodule.h>
#include "nnstreamer_log.h"
#include "nnstreamer_subplugin.h"
#include "nnstreamer_conf.h"
#include <nnstreamer_util.h>
```

Include dependency graph for nnstreamer_subplugin.c:



Enumerations

- enum subpluginSearchLogic { NNS_SEARCH_FILENAME, NNS_SEARCH_GETALL, NNS_SEARCH_NO_OP }

Functions

- static void init_subplugin (void)

  Create handles at the start of library.
- G_LOCK_DEFINE_STATIC (splock)

  Protects handles and subplugins.
- static void _spdata_destroy (gpointer _data)

  Private function for g_hash_table data destructor, GDestroyNotify.
- static subpluginData * _get_subplugin_data (subpluginType type, const gchar *name)

  Internal function to get sub-plugin data.
- static subpluginData * _search_subplugin (subpluginType type, const gchar *name, const gchar *path)

  Internal function to scan sub-plugin.
- const void * get_subplugin (subpluginType type, const char *name)

  Public function defined in the header.
- gchar ** get_all_subplugins (subpluginType type)

  Public function defined in the header.
- gboolean register_subplugin (subpluginType type, const char *name, const void *data)

  Public function defined in the header.
- gboolean unregister_subplugin (subpluginType type, const char *name)

  Public function defined in the header.
- static void _close_handle (gpointer data)

  dealloc function for handles
- void subplugin_set_custom_property_desc (subpluginType type, const char *name, const gchar *prop, va_list varargs)

  common interface to set custom property description of a sub-plugin.
- GData * subplugin_get_custom_property_desc (subpluginType type, const char *name)

  common interface to get custom property description of a sub-plugin.
- static void fini_subplugin (void)

  Free handles at the start of library.

Variables

- static GPtrArray * handles = NULL

  Array of dynamic loaded handles.
- subpluginData
- static GHashTable * subplugins [NNS_SUBPLUGIN_END] = { 0 }
- static subpluginSearchLogic searchAlgorithm [ ]

9.95.1 Detailed Description

Subplugin Manager for NNStreamer.

NNStreamer Subplugin Manager Copyright (C) 2018 MyungJoo Ham myungjoo.ham@samsung.com

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Library General Public License as published by the Free Software Foundation; version 2.1 of the License.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Library General Public License for more details.

Date

27 Nov 2018

See also

http://github.com/nnstreamer/nnstreamer

Author

MyungJoo Ham myungjoo.ham@samsung.com

Bug No known bugs except for NYI items

9.95.2 Enumeration Type Documentation

9.95.2.1 subpluginSearchLogic

```
enum subpluginSearchLogic
```

Enumerator

NNS_SEARCH_FILENAME	
NNS_SEARCH_GETALL	
NNS_SEARCH_NO_OP	

Definition at line 64 of file nnstreamer_subplugin.c.

9.95.3 Function Documentation

9.95.3.1 _close_handle()

```
static void _close_handle (
            gpointer data )  [static]
```

dealloc function for handles

Ubuntu 16.04 / GLIBC 2.23 Workaround If we do dlclose at exit() function, it may incur https://bugzilla.redhat.com/show_bug.cgi?id=1264556#c42 , which is a GLIBC bug at 2.23. The corresponding error message is: Inconsistency detected by ld.so: dl-close.c: 811: _dl_close: Assertion 'map->l_init_called' failed! Note that Tizen 5.5 / GLIBC 2.24 has the same bug!

Definition at line 305 of file nnstreamer_subplugin.c.

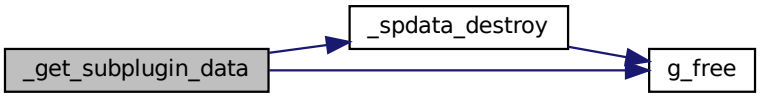
9.95.3.2 _get_subplugin_data()

```
static subpluginData* _get_subplugin_data (
            subpluginType type,
            const gchar * name )  [static]
```

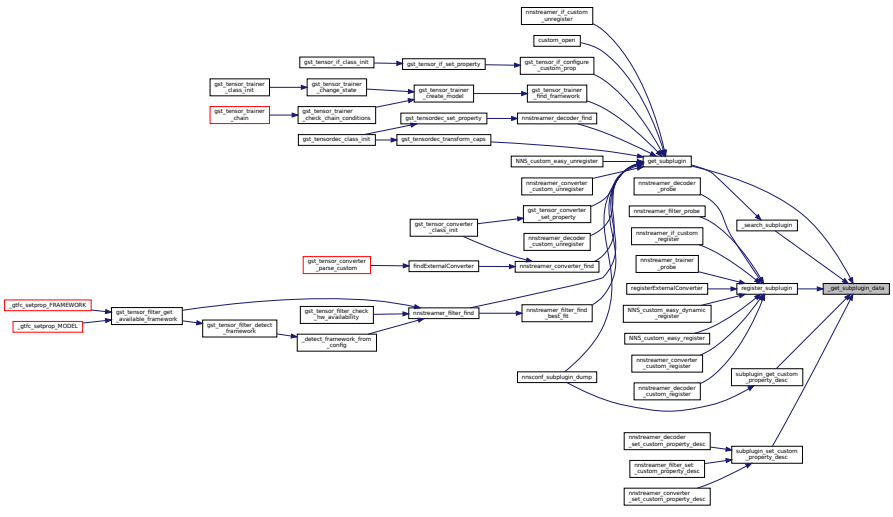
Internal function to get sub-plugin data.

Definition at line 87 of file nnstreamer_subplugin.c.

Here is the call graph for this function:



Here is the caller graph for this function:



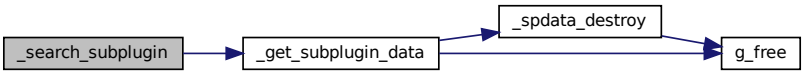
9.95.3.3 _search_subplugin()

```
static subpluginData* _search_subplugin (
            subpluginType type,
            const gchar * name,
            const gchar * path )  [static]
```

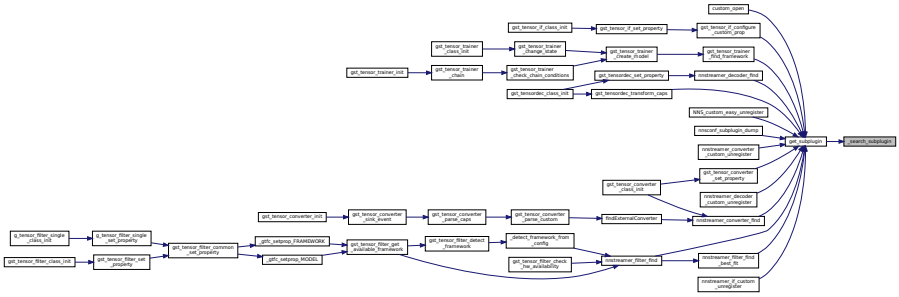
Internal function to scan sub-plugin.

Definition at line 108 of file nnstreamer_subplugin.c.

Here is the call graph for this function:



Here is the caller graph for this function:



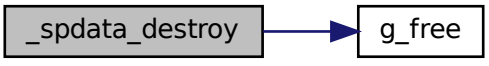
9.95.3.4 _spdata_destroy()

```
static void _spdata_destroy (
            gpointer _data )  [static]
```

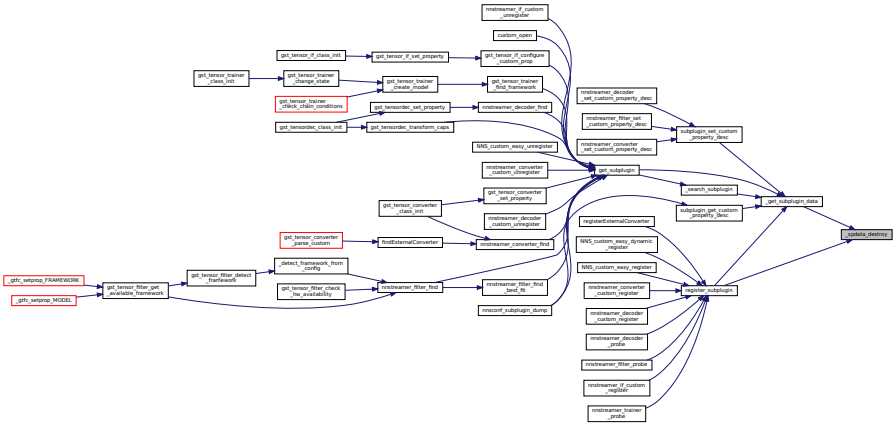
Private function for g_hash_table data destructor, GDestroyNotify.

Definition at line 54 of file nnstreamer_subplugin.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.95.3.5 fini_subplugin()

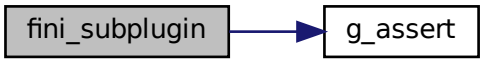
```
static void fini_subplugin (
              void  )  [static]
```

Free handles at the start of library.

Internal error (init not called?)

Definition at line 385 of file nnstreamer_subplugin.c.

Here is the call graph for this function:



9.95.3.6 G_LOCK_DEFINE_STATIC()

```
G_LOCK_DEFINE_STATIC (
              splock  )
```

Protects handles and subplugins.

9.95.3.7 get_all_subplugins()

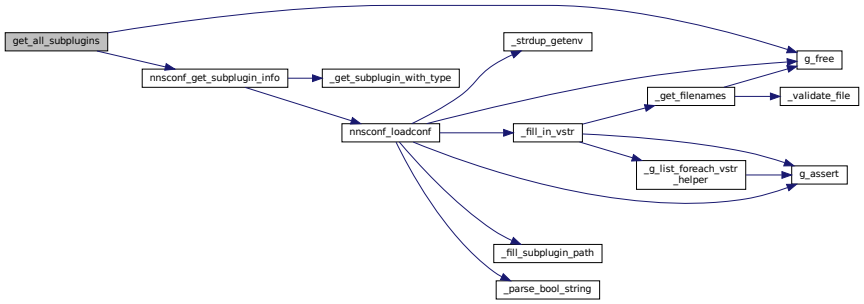
```
gchar** get_all_subplugins (
            subpluginType type )
```

Public function defined in the header.

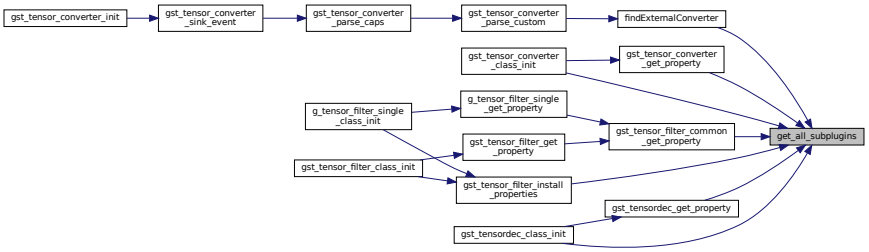
Get the list of registered subplugins.

Definition at line 176 of file nnstreamer_subplugin.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.95.3.8 get_subplugin()

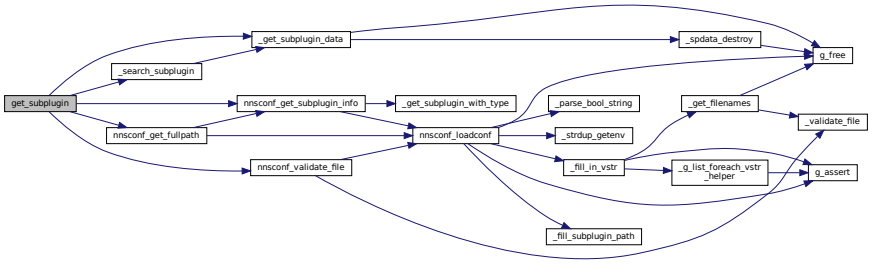
```
const void* get_subplugin (
            subpluginType type,
            const char * name )
```

Public function defined in the header.

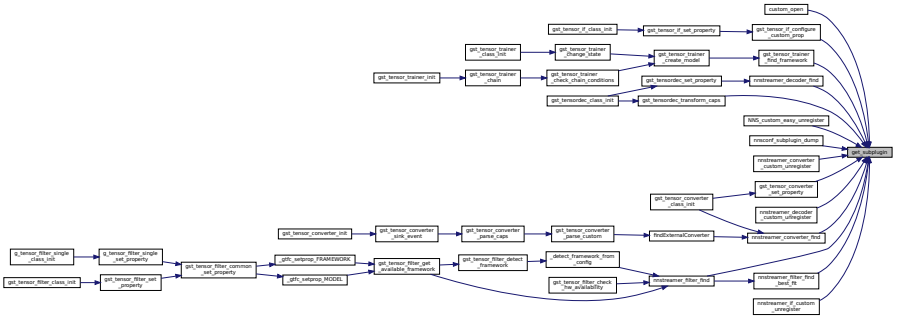
Retrieve the registered data with the subplugin name. Search and register if found with the conf

Definition at line 141 of file nnstreamer_subplugin.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.95.3.9 init_subplugin()

```
static void init_subplugin (
            void  )  [static]
```

Create handles at the start of library.

< The name of subplugin

< subplugin specific data forwarded from the subplugin

< [OPTIONAL] subplugin specific custom property desc list

Internal error (duplicated init call?)

Definition at line 37 of file nnstreamer_subplugin.c.

9.95.3.10 register_subplugin()

```
gboolean register_subplugin (
             subpluginType type,
             const char * name,
             const void * data )
```

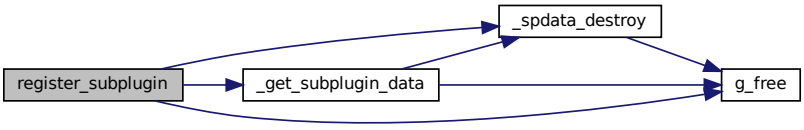
Public function defined in the header.

Register the subplugin. If duplicated name exists, it is rejected.

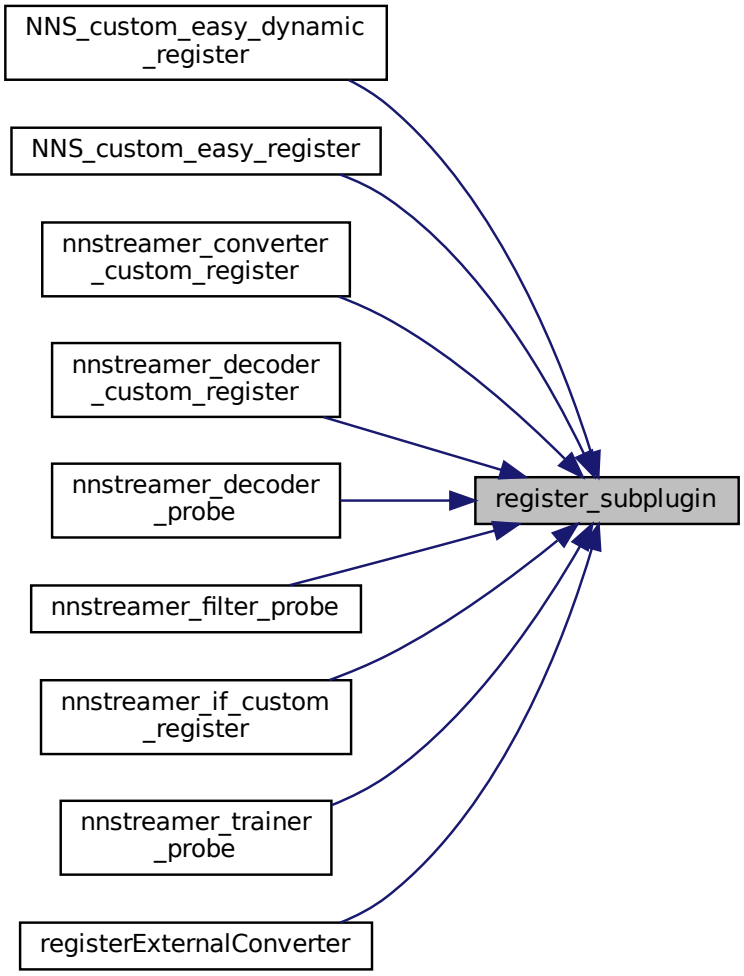
Todo data out of scope at add

Definition at line 225 of file nnstreamer_subplugin.c.

Here is the call graph for this function:



Here is the caller graph for this function:



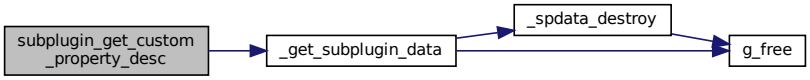
9.95.3.11 subplugin_get_custom_property_desc()

```
GData* subplugin_get_custom_property_desc (
            subpluginType type,
            const char * name )
```

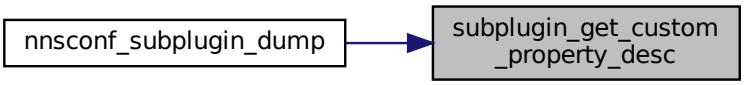
common interface to get custom property description of a sub-plugin.

Definition at line 359 of file nnstreamer_subplugin.c.

Here is the call graph for this function:



Here is the caller graph for this function:



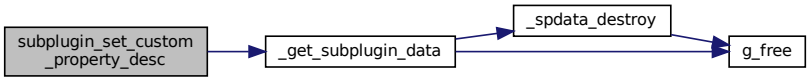
9.95.3.12 subplugin_set_custom_property_desc()

```
void subplugin_set_custom_property_desc (
            subpluginType type,
            const char * name,
            const gchar * prop,
            va_list varargs )
```

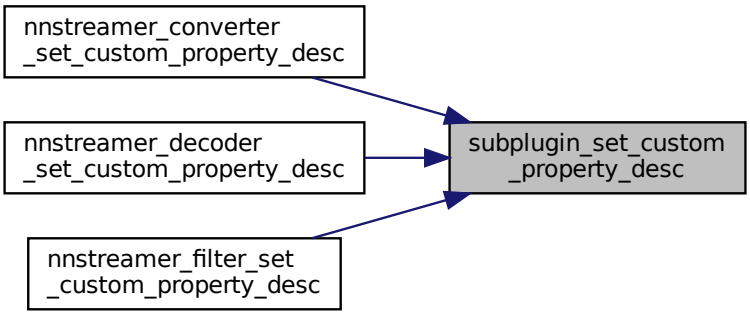
common interface to set custom property description of a sub-plugin.

Definition at line 329 of file nnstreamer_subplugin.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.95.3.13 unregister_subplugin()

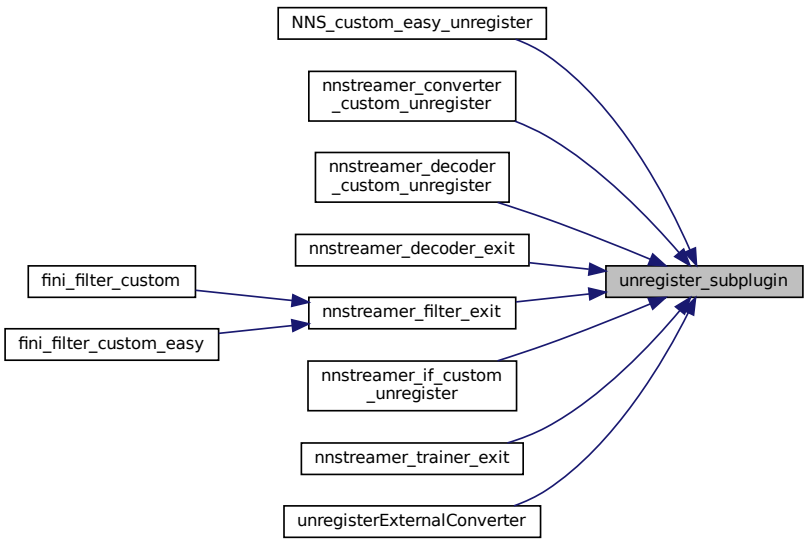
```
gboolean unregister_subplugin (
            subpluginType type,
            const char * name )
```

Public function defined in the header.

Unregister the subplugin.

Definition at line 289 of file nnstreamer_subplugin.c.

Here is the caller graph for this function:



9.95.4 Variable Documentation

9.95.4.1 handles

```
GPtrArray* handles = NULL  [static]
```

Array of dynamic loaded handles.

Definition at line 35 of file nnstreamer_subplugin.c.

9.95.4.2 searchAlgorithm

```
subpluginSearchLogic searchAlgorithm[]  [static]
```

Initial value:

```
= {
  [NNS_SUBPLUGIN_FILTER] = NNS_SEARCH_FILENAME,
  [NNS_SUBPLUGIN_DECODER] = NNS_SEARCH_FILENAME,
  [NNS_EASY_CUSTOM_FILTER] = NNS_SEARCH_FILENAME,
  [NNS_SUBPLUGIN_CONVERTER] = NNS_SEARCH_GETALL,
  [NNS_SUBPLUGIN_TRAINER] = NNS_SEARCH_FILENAME,
  [NNS_CUSTOM_CONVERTER] = NNS_SEARCH_NO_OP,
  [NNS_CUSTOM_DECODER] = NNS_SEARCH_NO_OP,
  [NNS_IF_CUSTOM] = NNS_SEARCH_NO_OP,
  [NNS_SUBPLUGIN_END] = NNS_SEARCH_NO_OP,
}
```

Definition at line 71 of file nnstreamer_subplugin.c.

9.95.4.3 subpluginData

```
subpluginData
```

Definition at line 45 of file nnstreamer_subplugin.c.

9.95.4.4 subplugins

```
GHashTable* subplugins[NNS_SUBPLUGIN_END] = { 0 }  [static]
```

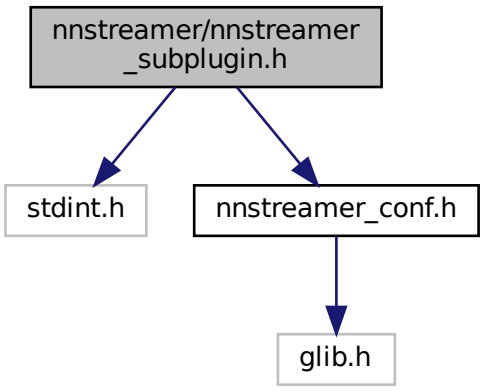
Definition at line 47 of file nnstreamer_subplugin.c.

9.96 nnstreamer/nnstreamer_subplugin.h File Reference

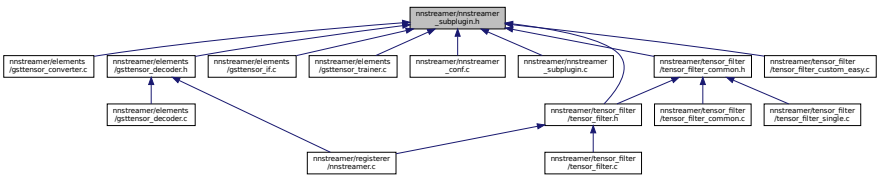
Subplugin Manager for NNStreamer.

```
#include <stdint.h>
#include "nnstreamer_conf.h"
```

Include dependency graph for nnstreamer_subplugin.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define NNS_SUBPLUGIN_CHECKER (0xdeadbeef)

Enumerations

- enum subpluginType {
  NNS_SUBPLUGIN_FILTER = NNSCONF_PATH_FILTERS, NNS_SUBPLUGIN_DECODER = NNSCON↩
  F_PATH_DECODERS, NNS_EASY_CUSTOM_FILTER = NNSCONF_PATH_EASY_CUSTOM_FILTERS,
  NNS_SUBPLUGIN_CONVERTER = NNSCONF_PATH_CONVERTERS,
  NNS_SUBPLUGIN_TRAINER = NNSCONF_PATH_TRAINERS, NNS_CUSTOM_CONVERTER, NNS_CUSTOM_DECODER,
  NNS_IF_CUSTOM,
  NNS_SUBPLUGIN_END }

Functions

- const void * get_subplugin (subpluginType type, const char *name)

  Retrieve the registered data with the subplugin name.
- gchar ** get_all_subplugins (subpluginType type)

  Get the list of registered subplugins.
- gboolean register_subplugin (subpluginType type, const char *name, const void *data)

  Register the subplugin. If duplicated name exists, it is rejected.
- gboolean unregister_subplugin (subpluginType type, const char *name)

  Unregister the subplugin.
- void subplugin_set_custom_property_desc (subpluginType type, const char *name, const gchar *prop, va_list varargs)

  common interface to set custom property description of a sub-plugin.
- GData * subplugin_get_custom_property_desc (subpluginType type, const char *name)

  common interface to get custom property description of a sub-plugin.

9.96.1 Detailed Description

Subplugin Manager for NNStreamer.

NNStreamer Subplugin Manager Copyright (C) 2018 MyungJoo Ham myungjoo.ham@samsung.com

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Library General Public License as published by the Free Software Foundation; version 2.1 of the License.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Library General Public License for more details.

Date

27 Nov 2018

See also

http://github.com/nnstreamer/nnstreamer

Author

MyungJoo Ham myungjoo.ham@samsung.com

Bug No known bugs except for NYI items

To Packagers:

This file is to be packaged as "devel" package for NN developers. (subplugin writers)

Note

Any independent subplugin (existing as an independent .so) should call register_subplugin () (or its wrapper) with the subplugin's constructor function.

9.96.2 Macro Definition Documentation

9.96.2.1 NNS_SUBPLUGIN_CHECKER

```
#define NNS_SUBPLUGIN_CHECKER (0xdeadbeef)
```

Definition at line 53 of file nnstreamer_subplugin.h.

9.96.3 Enumeration Type Documentation

9.96.3.1 subpluginType

```
enum subpluginType
```

Enumerator

NNS_SUBPLUGIN_FILTER	
NNS_SUBPLUGIN_DECODER	
NNS_EASY_CUSTOM_FILTER	
NNS_SUBPLUGIN_CONVERTER	
NNS_SUBPLUGIN_TRAINER	
NNS_CUSTOM_CONVERTER	
NNS_CUSTOM_DECODER	
NNS_IF_CUSTOM	
NNS_SUBPLUGIN_END	

Definition at line 40 of file nnstreamer_subplugin.h.

9.96.4 Function Documentation

9.96.4.1 get_all_subplugins()

```
gchar** get_all_subplugins (
            subpluginType type )
```

Get the list of registered subplugins.

Parameters

in	*type*	Subplugin Type

Returns

The list of subplugin name

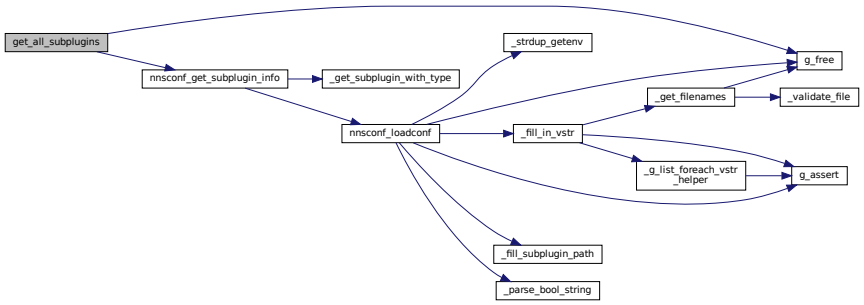
Note

Caller should free the returned value using g_strfreev()

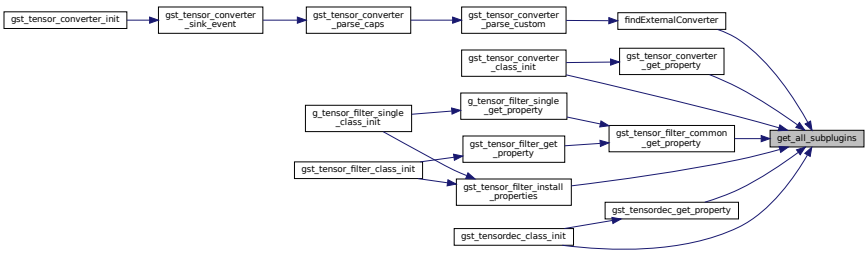
Get the list of registered subplugins.

Definition at line 176 of file nnstreamer_subplugin.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.96.4.2 get_subplugin()

```
const void* get_subplugin (
            subpluginType type,
            const char * name )
```

Retrieve the registered data with the subplugin name.

Parameters

in	*type*	Subplugin Type
in	*name*	Subplugin Name. The filename should be libnnstreamer_${type}_${name}.so

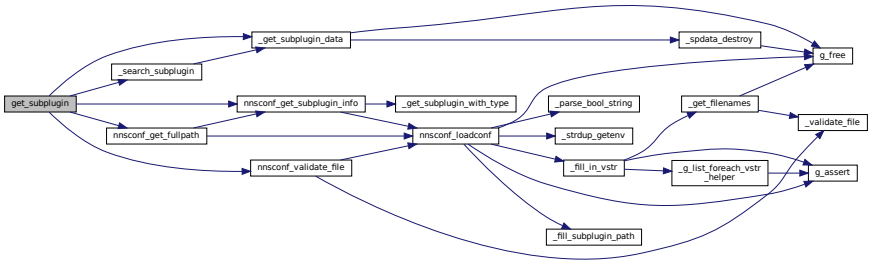
Returns

The registered data

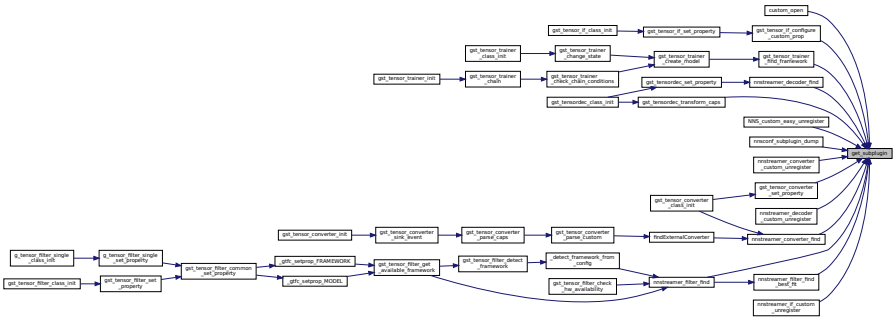
Retrieve the registered data with the subplugin name. Search and register if found with the conf

Definition at line 141 of file nnstreamer_subplugin.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.96.4.3 register_subplugin()

```
gboolean register_subplugin (
            subpluginType type,
            const char * name,
            const void * data )
```

Register the subplugin. If duplicated name exists, it is rejected.

Parameters

in	*type*	Subplugin Type
in	*name*	Subplugin Name. The filename should be subplugin_prefixes[type]${name}.so
in	*data*	The registered data

Returns

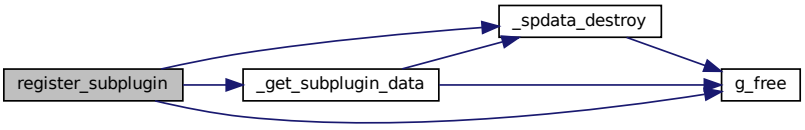
TRUE if registered as new. FALSE if duplicated (overwritten/updated).

Register the subplugin. If duplicated name exists, it is rejected.

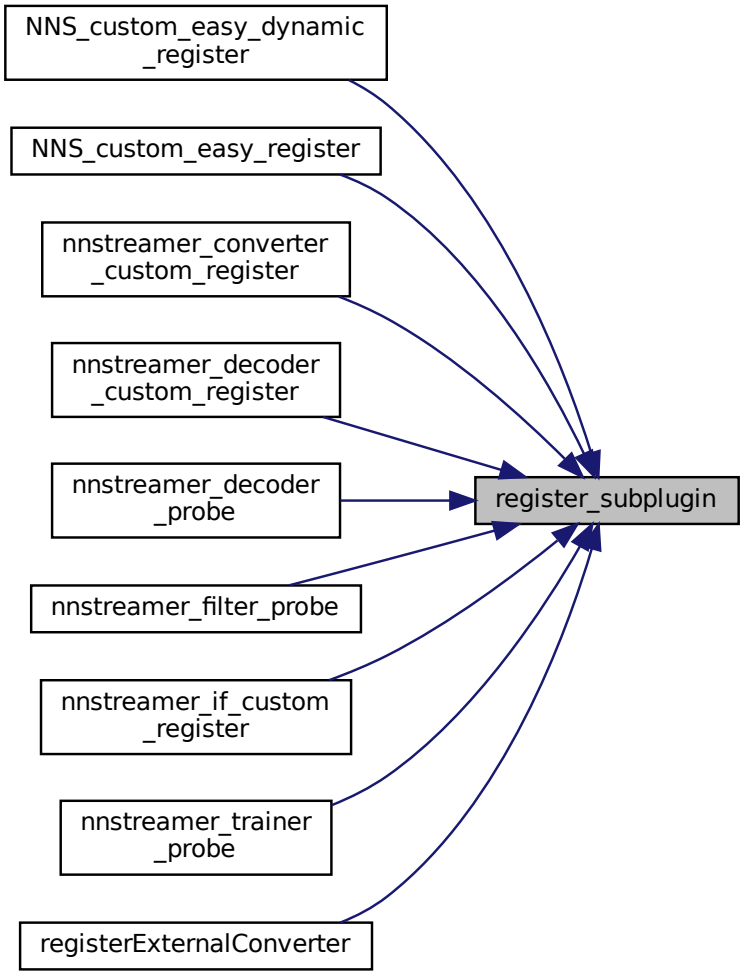
Todo data out of scope at add

Definition at line 225 of file nnstreamer_subplugin.c.

Here is the call graph for this function:



Here is the caller graph for this function:



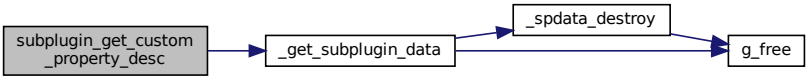
9.96.4.4 subplugin_get_custom_property_desc()

```
GData* subplugin_get_custom_property_desc (
            subpluginType type,
            const char * name )
```

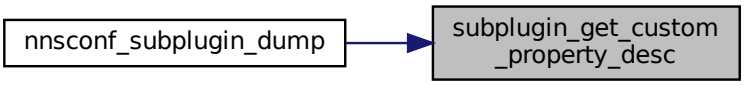
common interface to get custom property description of a sub-plugin.

Definition at line 359 of file nnstreamer_subplugin.c.

Here is the call graph for this function:



Here is the caller graph for this function:



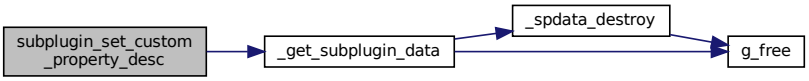
9.96.4.5 subplugin_set_custom_property_desc()

```
void subplugin_set_custom_property_desc (
            subpluginType type,
            const char * name,
            const gchar * prop,
            va_list varargs )
```

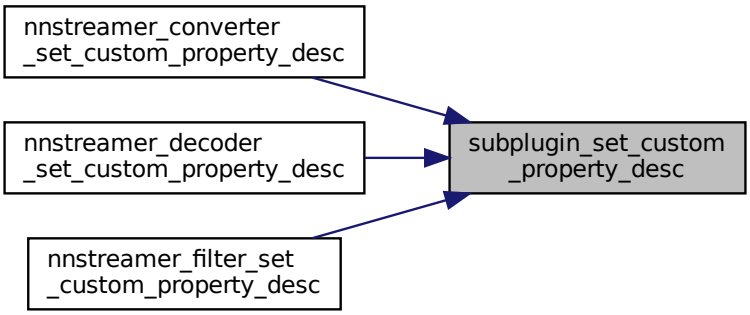
common interface to set custom property description of a sub-plugin.

Definition at line 329 of file nnstreamer_subplugin.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.96.4.6 unregister_subplugin()

```
gboolean unregister_subplugin (
            subpluginType type,
            const char * name )
```

Unregister the subplugin.

Parameters

in	*type*	Subplugin type
in	*name*	Subplugin Name. The filename should be subplugin_prefixes[type]${name}.so

Returns

TRUE if unregistered. FALSE if rejected or error.

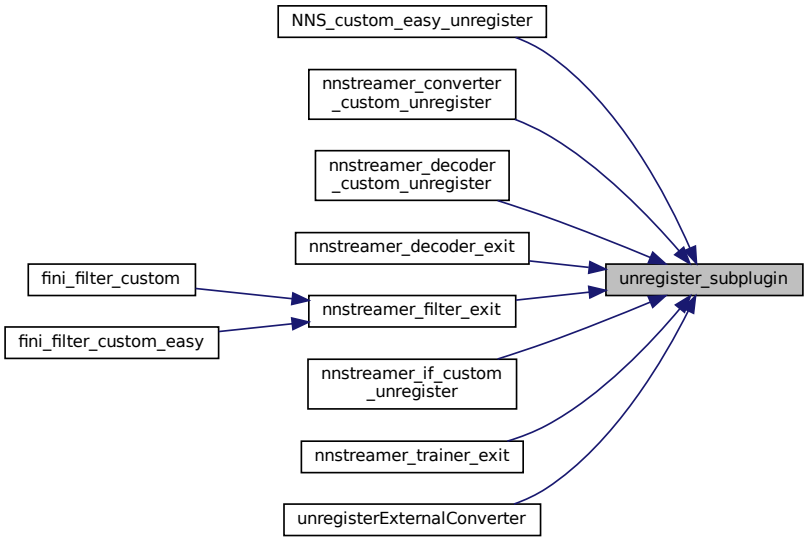
Warning

Subplugins checked out with get_subplugins can still be used after unregister.

Unregister the subplugin.

Definition at line 289 of file nnstreamer_subplugin.c.

Here is the caller graph for this function:

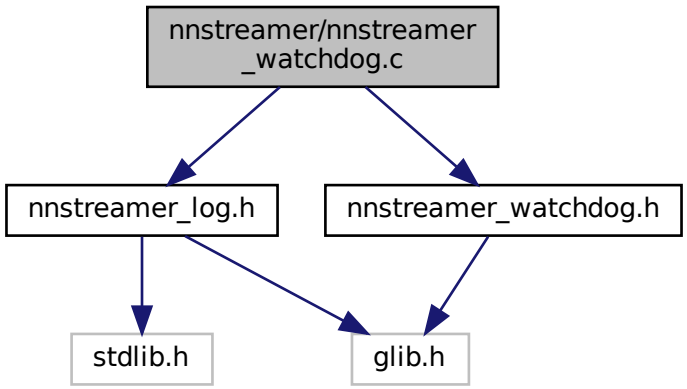


9.97 nnstreamer/nnstreamer_watchdog.c File Reference

NNStreamer watchdog to manage the schedule in the element.

```
#include <nnstreamer_log.h>
#include "nnstreamer_watchdog.h"
```

Include dependency graph for nnstreamer_watchdog.c:



Classes

- struct _NnstWatchdog

  Structure for NNStreamer watchdog.

Typedefs

- typedef struct _NnstWatchdog NnstWatchdog

  Structure for NNStreamer watchdog.

Functions

- static gboolean _loop_running_cb (NnstWatchdog *watchdog)

  Called when loop is running.
- static gpointer _nnstreamer_watchdog_thread (gpointer ptr)

  Watchdog thread.
- gboolean nnstreamer_watchdog_create (nns_watchdog_h *watchdog_h)

  Create nnstreamer watchdog. Recommended using watchdog handle with proper lock (e.g., GST_OBJECT_LOCK())
- void nnstreamer_watchdog_destroy (nns_watchdog_h watchdog_h)

  Destroy watchdog source. Recommended using watchdog handle with proper lock (e.g., GST_OBJECT_LOCK())
- void nnstreamer_watchdog_release (nns_watchdog_h watchdog_h)

  Release watchdog source. Recommended using watchdog handle with proper lock (e.g., GST_OBJECT_LOCK())
- gboolean nnstreamer_watchdog_feed (nns_watchdog_h watchdog_h, GSourceFunc func, guint interval, void *user_data)

  Set watchdog timeout. Recommended using watchdog handle with proper lock (e.g., GST_OBJECT_LOCK())

9.97.1 Detailed Description

NNStreamer watchdog to manage the schedule in the element.

NNStreamer watchdog Copyright (C) 2024 Gichan Jang gichan2.jang@samsung.com

Date

30 Oct 2024

See also

https://github.com/nnstreamer/nnstreamer

Author

Gichan Jang gichan2.jang@samsung.com

Bug No known bugs except for NYI items

9.97.2 Typedef Documentation

9.97.2.1 NnstWatchdog

```
typedef struct _NnstWatchdog NnstWatchdog
```

Structure for NNStreamer watchdog.

9.97.3 Function Documentation

9.97.3.1 _loop_running_cb()

```
static gboolean _loop_running_cb (
            NnstWatchdog * watchdog )  [static]
```

Called when loop is running.

Definition at line 35 of file nnstreamer_watchdog.c.

Here is the caller graph for this function:

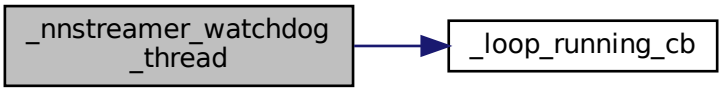
9.97.3.2 _nnstreamer_watchdog_thread()

```
static gpointer _nnstreamer_watchdog_thread (
            gpointer ptr )  [static]
```

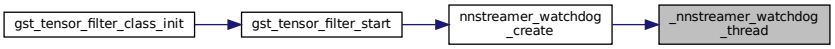
Watchdog thread.

Definition at line 48 of file nnstreamer_watchdog.c.

Here is the call graph for this function:



Here is the caller graph for this function:



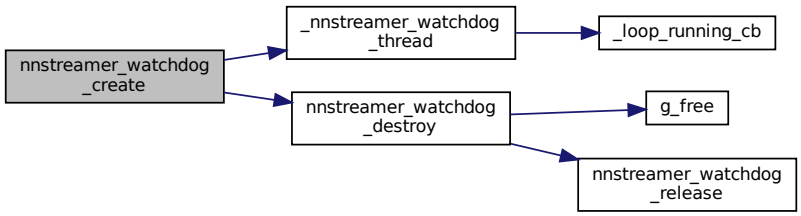
9.97.3.3 nnstreamer_watchdog_create()

```
gboolean nnstreamer_watchdog_create (
            nns_watchdog_h * watchdog_h )
```

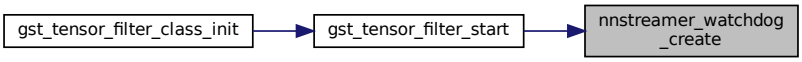
Create nnstreamer watchdog. Recommended using watchdog handle with proper lock (e.g., GST_OBJECT_LO↩ CK())

Definition at line 72 of file nnstreamer_watchdog.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.97.3.4 nnstreamer_watchdog_destroy()

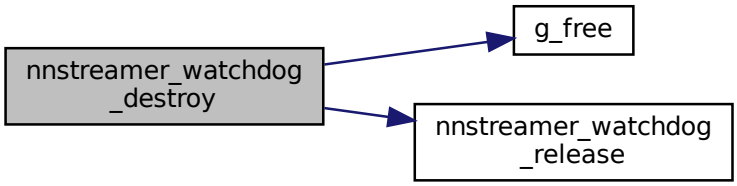
```
void nnstreamer_watchdog_destroy (
            nns_watchdog_h watchdog_h )
```

Destroy watchdog source. Recommended using watchdog handle with proper lock (e.g., GST_OBJECT_LOCK())

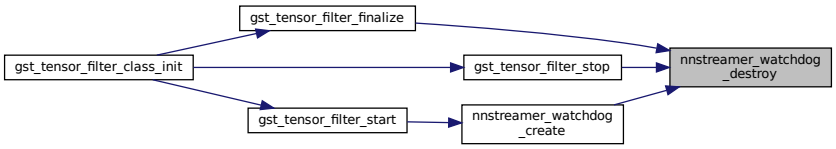
Destroy nnstreamer watchdog. Recommended using watchdog handle with proper lock (e.g., GST_OBJECT_LO↩ CK())

Definition at line 133 of file nnstreamer_watchdog.c.

Here is the call graph for this function:



Here is the caller graph for this function:



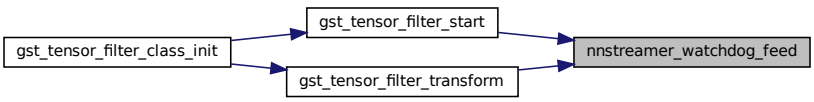
9.97.3.5 nnstreamer_watchdog_feed()

```
gboolean nnstreamer_watchdog_feed (
            nns_watchdog_h watchdog_h,
            GSourceFunc func,
            guint interval,
            void * user_data )
```

Set watchdog timeout. Recommended using watchdog handle with proper lock (e.g., GST_OBJECT_LOCK())

Definition at line 171 of file nnstreamer_watchdog.c.

Here is the caller graph for this function:



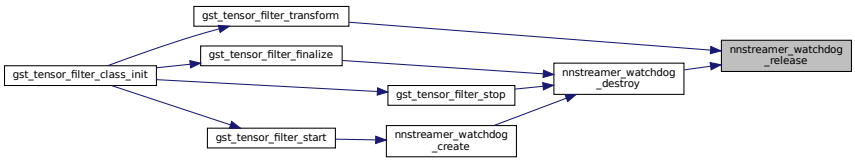
9.97.3.6 nnstreamer_watchdog_release()

```
void nnstreamer_watchdog_release (
            nns_watchdog_h watchdog_h )
```

Release watchdog source. Recommended using watchdog handle with proper lock (e.g., GST_OBJECT_LOCK())

Definition at line 157 of file nnstreamer_watchdog.c.

Here is the caller graph for this function:

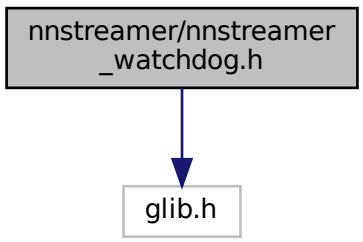


9.98 nnstreamer/nnstreamer_watchdog.h File Reference

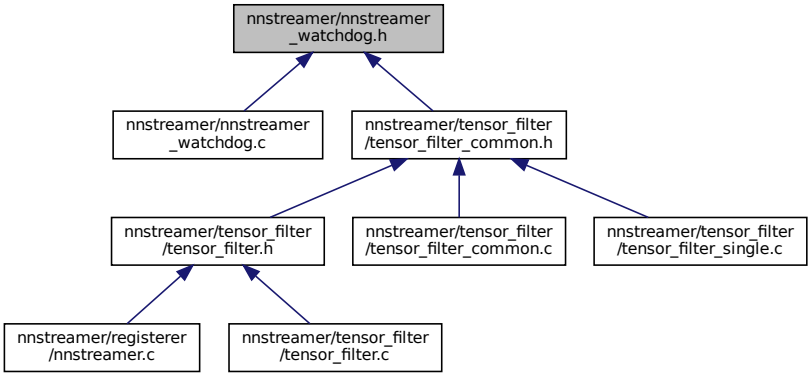
NNStreamer watchdog header to manage the schedule in the element.

```
#include <glib.h>
```

Include dependency graph for nnstreamer_watchdog.h:



This graph shows which files directly or indirectly include this file:



Typedefs

- typedef void * nns_watchdog_h

Functions

- gboolean nnstreamer_watchdog_create (nns_watchdog_h *watchdog_h)

  Create nnstreamer watchdog. Recommended using watchdog handle with proper lock (e.g., GST_OBJECT_LOCK())
- void nnstreamer_watchdog_destroy (nns_watchdog_h watchdog_h)

  Destroy nnstreamer watchdog. Recommended using watchdog handle with proper lock (e.g., GST_OBJECT_LO↩ CK())
- void nnstreamer_watchdog_release (nns_watchdog_h watchdog_h)

  Release watchdog source. Recommended using watchdog handle with proper lock (e.g., GST_OBJECT_LOCK())
- gboolean nnstreamer_watchdog_feed (nns_watchdog_h watchdog_h, GSourceFunc func, guint interval, void *user_data)

  Set watchdog timeout. Recommended using watchdog handle with proper lock (e.g., GST_OBJECT_LOCK())

9.98.1 Detailed Description

NNStreamer watchdog header to manage the schedule in the element.

NNStreamer watchdog header Copyright (C) 2024 Gichan Jang gichan2.jang@samsung.com

Date

30 Oct 2024

See also

https://github.com/nnstreamer/nnstreamer

Author

Gichan Jang gichan2.jang@samsung.com

Bug No known bugs except for NYI items

9.98.2 Typedef Documentation

9.98.2.1 nns_watchdog_h

```
typedef void* nns_watchdog_h
```

Definition at line 25 of file nnstreamer_watchdog.h.

9.98.3 Function Documentation

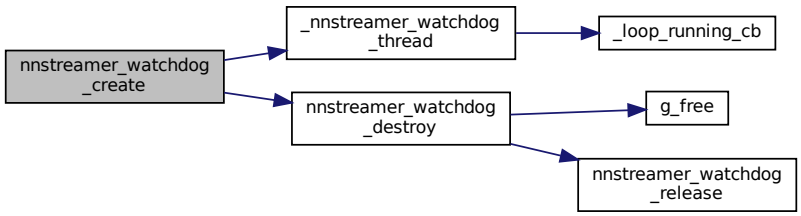
9.98.3.1 nnstreamer_watchdog_create()

```
gboolean nnstreamer_watchdog_create (
            nns_watchdog_h * watchdog_h )
```

Create nnstreamer watchdog. Recommended using watchdog handle with proper lock (e.g., GST_OBJECT_LO↩ CK())

Definition at line 72 of file nnstreamer_watchdog.c.

Here is the call graph for this function:



Here is the caller graph for this function:

9.98.3.2 nnstreamer_watchdog_destroy()

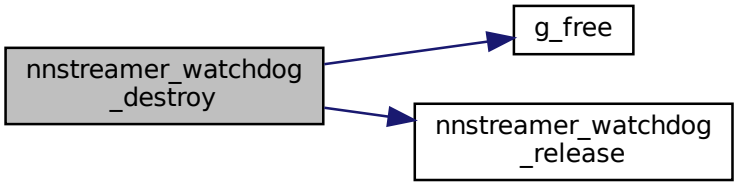
```
void nnstreamer_watchdog_destroy (
            nns_watchdog_h watchdog_h )
```

Destroy nnstreamer watchdog. Recommended using watchdog handle with proper lock (e.g., GST_OBJECT_LO↩ CK())

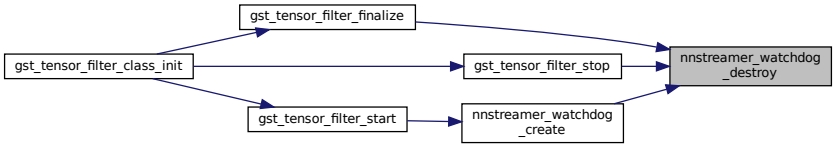
Destroy nnstreamer watchdog. Recommended using watchdog handle with proper lock (e.g., GST_OBJECT_LO↩ CK())

Definition at line 133 of file nnstreamer_watchdog.c.

Here is the call graph for this function:



Here is the caller graph for this function:



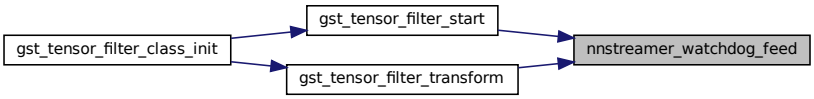
9.98.3.3 nnstreamer_watchdog_feed()

```
gboolean nnstreamer_watchdog_feed (
            nns_watchdog_h watchdog_h,
            GSourceFunc func,
            guint interval,
            void * user_data )
```

Set watchdog timeout. Recommended using watchdog handle with proper lock (e.g., GST_OBJECT_LOCK())

Definition at line 171 of file nnstreamer_watchdog.c.

Here is the caller graph for this function:



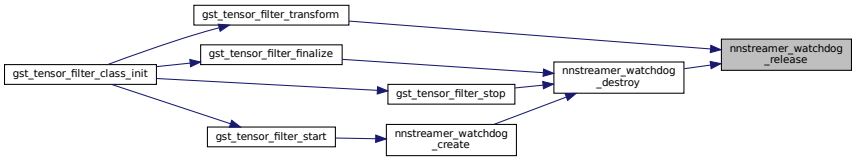
9.98.3.4 nnstreamer_watchdog_release()

```
void nnstreamer_watchdog_release (
            nns_watchdog_h watchdog_h )
```

Release watchdog source. Recommended using watchdog handle with proper lock (e.g., GST_OBJECT_LOCK())

Definition at line 157 of file nnstreamer_watchdog.c.

Here is the caller graph for this function:



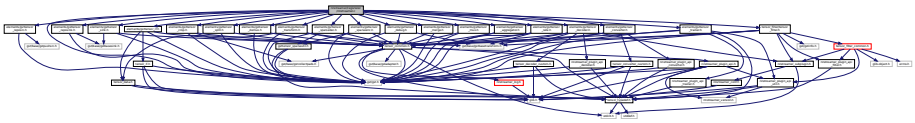
9.99 nnstreamer/registerer/nnstreamer.c File Reference

Registers all nnstreamer plugins for gstreamer so that we can have a single big binary.

```
#include <gst/gst.h>
#include <elements/gsttensor_aggregator.h>
#include <elements/gsttensor_converter.h>
#include <elements/gsttensor_crop.h>
#include <elements/gsttensor_debug.h>
#include <elements/gsttensor_decoder.h>
#include <elements/gsttensor_demux.h>
#include <elements/gsttensor_if.h>
#include <elements/gsttensor_merge.h>
#include <elements/gsttensor_mux.h>
#include <elements/gsttensor_rate.h>
#include <elements/gsttensor_reposink.h>
#include <elements/gsttensor_reposrc.h>
#include <elements/gsttensor_sink.h>
#include <elements/gsttensor_sparsedec.h>
#include <elements/gsttensor_sparseenc.h>
```

```
#include <elements/gsttensor_split.h>
#include <elements/gsttensor_transform.h>
#include <elements/gsttensor_trainer.h>
#include <tensor_filter/tensor_filter.h>
```

Include dependency graph for nnstreamer.c:



Macros

- #define NNSTREAMER_INIT(plugin, name, type)
- #define PACKAGE "nnstreamer"

Functions

- GST_ERROR ("Failed to register nnstreamer plugin : tensor_" # name)

  *type)) { *
- while (0)

  Function to initialize all nnstreamer elements.
- GST_PLUGIN_DEFINE (GST_VERSION_MAJOR, GST_VERSION_MINOR, nnstreamer, "NNStreamer plugin library allows neural networks in GStreamer pipelines. Use nnstreamer-check utility for more information of the current NNStreamer installation.", gst_nnstreamer_init, VERSION, "LGPL", "nnstreamer", "https↩ ://github.com/nnstreamer/nnstreamer")

Variables

- return FALSE

9.99.1 Detailed Description

Registers all nnstreamer plugins for gstreamer so that we can have a single big binary.

Date

11 Oct 2018

See also

https://github.com/nnstreamer/nnstreamer

Author

MyungJoo Ham myungjoo.ham@samsung.com

Bug No known bugs except for NYI items

9.99.2 Macro Definition Documentation

9.99.2.1 NNSTREAMER_INIT

```
#define NNSTREAMER_INIT(
              plugin,
              name,
              type )
```

Value:

```
  do { \
    if (!gst_element_register (plugin, "tensor_" # name, GST_RANK_NONE, GST_TYPE_TENSOR_
```

Definition at line 80 of file nnstreamer.c.

9.99.2.2 PACKAGE

```
#define PACKAGE "nnstreamer"
```

Definition at line 125 of file nnstreamer.c.

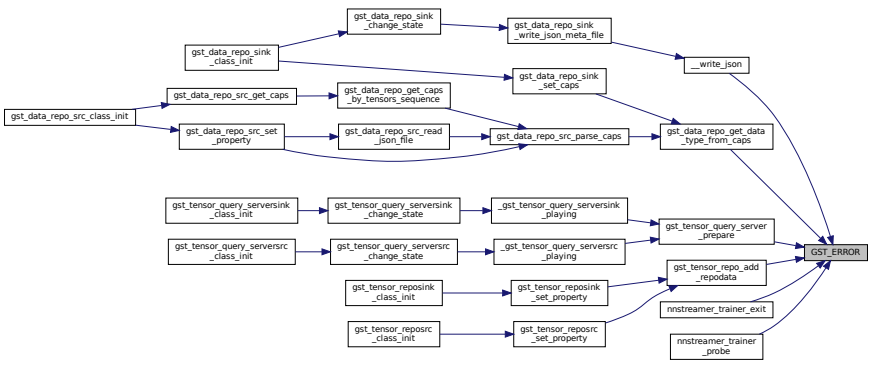
9.99.3 Function Documentation

9.99.3.1 GST_ERROR()

```
GST_ERROR (
            "Failed to register nnstreamer plugin :  tensor_" # name )
```

type)) { \

Here is the caller graph for this function:



9.99.3.2 GST_PLUGIN_DEFINE()

```
GST_PLUGIN_DEFINE (
            GST_VERSION_MAJOR ,
            GST_VERSION_MINOR ,
            nnstreamer ,
            "NNStreamer plugin library allows neural networks in GStreamer pipelines.  Use
nnstreamer-check utility for more information of the current NNStreamer installation." ,
            gst_nnstreamer_init ,
            VERSION ,
            "LGPL" ,
            "nnstreamer" ,
            "https://github.com/nnstreamer/nnstreamer"  )
```

9.99.3.3 while()

```
while (
            0  )
```

Function to initialize all nnstreamer elements.

Definition at line 86 of file nnstreamer.c.

9.99.4 Variable Documentation

9.99.4.1 FALSE

```
return FALSE
```

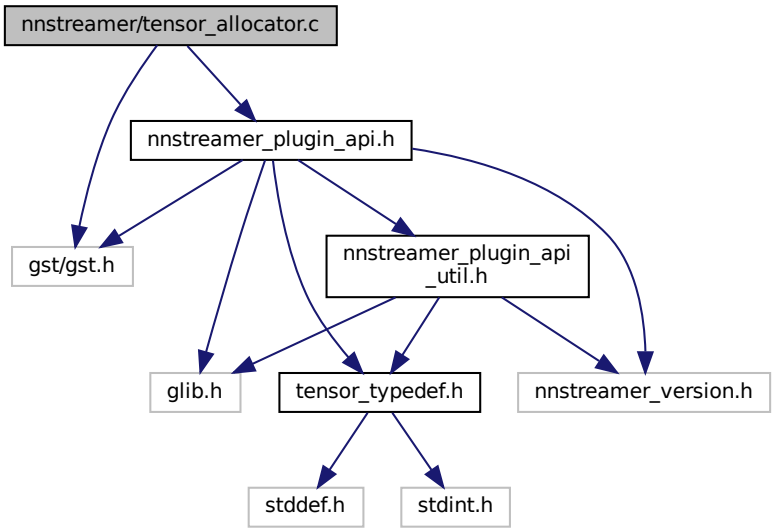
Definition at line 84 of file nnstreamer.c.

9.100 nnstreamer/tensor_allocator.c File Reference

Allocator for memory alignment.

```
#include <gst/gst.h>
#include "nnstreamer_plugin_api.h"
```

Include dependency graph for tensor_allocator.c:



Classes

- struct GstTensorAllocator

  struct for type GstTensorAllocator
- struct GstTensorAllocatorClass

  struct for class GstTensorAllocatorClass

Macros

- #define GST_TENSOR_ALLOCATOR "GstTensorAllocator"

Functions

- static GType gst_tensor_allocator_get_type (void)
- G_DEFINE_TYPE (GstTensorAllocator, gst_tensor_allocator, GST_TYPE_ALLOCATOR)
- static GstMemory * _alloc (GstAllocator *allocator, gsize size, GstAllocationParams *params)

  allocation wrapper that binds alignment parameter
- static void gst_tensor_allocator_class_init (GstTensorAllocatorClass *klass)

  class initization for GstTensorAllocatorClass
- static void gst_tensor_allocator_init (GstTensorAllocator *allocator)

  initialization for GstTensorAllocator
- void gst_tensor_alloc_init (gsize alignment)

  set alignment that default allocator would align to

Variables

- static gsize gst_tensor_allocator_alignment = 0

9.100.1 Detailed Description

Allocator for memory alignment.

Copyright (C) 2021 Junhwan Kim jejudo.kim@samsung.com

Date

12 May 2021

Author

Junhwan Kim jejudo.kim@samsung.com

See also

http://github.com/nnstreamer/nnstreamer

Bug No known bugs

9.100.2 Macro Definition Documentation

9.100.2.1 GST_TENSOR_ALLOCATOR

```
#define GST_TENSOR_ALLOCATOR "GstTensorAllocator"
```

Definition at line 17 of file tensor_allocator.c.

9.100.3 Function Documentation

9.100.3.1 _alloc()

```
static GstMemory* _alloc (
            GstAllocator * allocator,
            gsize size,
            GstAllocationParams * params )  [static]
```

allocation wrapper that binds alignment parameter

Definition at line 45 of file tensor_allocator.c.

Here is the caller graph for this function:

9.100.3.2 G_DEFINE_TYPE()

```
G_DEFINE_TYPE (
            GstTensorAllocator ,
            gst_tensor_allocator ,
            GST_TYPE_ALLOCATOR  )
```

9.100.3.3 gst_tensor_alloc_init()

```
void gst_tensor_alloc_init (
            gsize alignment )
```

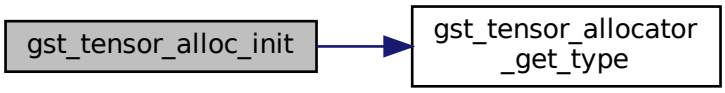
set alignment that default allocator would align to

Parameters

alignment	bytes of alignment

Definition at line 109 of file tensor_allocator.c.

Here is the call graph for this function:



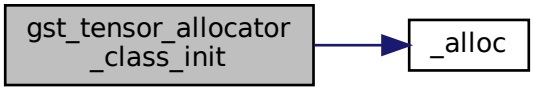
9.100.3.4 gst_tensor_allocator_class_init()

```
static void gst_tensor_allocator_class_init (
            GstTensorAllocatorClass * klass )  [static]
```

class initization for GstTensorAllocatorClass

Definition at line 68 of file tensor_allocator.c.

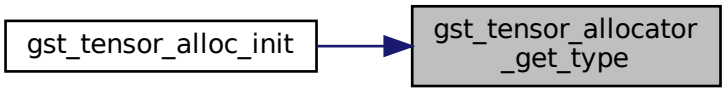
Here is the call graph for this function:



9.100.3.5 gst_tensor_allocator_get_type()

```
static GType gst_tensor_allocator_get_type (
            void  )  [static]
```

Here is the caller graph for this function:



9.100.3.6 gst_tensor_allocator_init()

```
static void gst_tensor_allocator_init (
            GstTensorAllocator * allocator )  [static]
```

initialization for GstTensorAllocator

Definition at line 87 of file tensor_allocator.c.

9.100.4 Variable Documentation

9.100.4.1 gst_tensor_allocator_alignment

```
gsize gst_tensor_allocator_alignment = 0  [static]
```

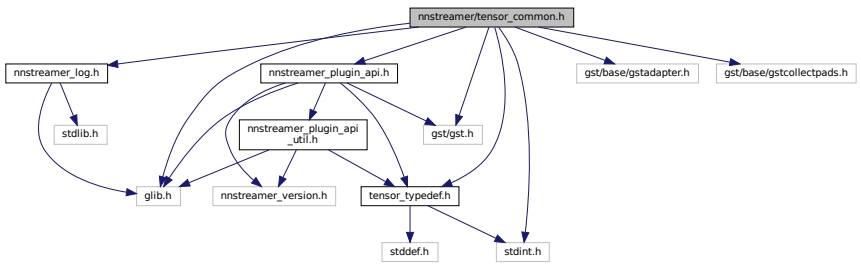
Definition at line 19 of file tensor_allocator.c.

9.101 nnstreamer/tensor_common.h File Reference

Common header file for NNStreamer, the GStreamer plugin for neural networks.

```
#include <glib.h>
#include <stdint.h>
#include <gst/gst.h>
#include <gst/base/gstadapter.h>
#include <gst/base/gstcollectpads.h>
#include "tensor_typedef.h"
#include "nnstreamer_log.h"
#include "nnstreamer_plugin_api.h"
```

Include dependency graph for tensor_common.h:



This graph shows which files directly or indirectly include this file:

Classes

- struct _tensor_sync_basepad_data

  Tensor Merge/Mux sync data for baspad mode.
- struct _tensor_time_sync_data

  Tensor Merge/Mux time sync data.
- struct GstTensorCollectPadData

  Internal data structure for Collect Pad in mux / merge.
- struct GstTensorPad

  Internal data structure for pad in demux / split.

Macros

- #define nns_memcpy memcpy
- #define nns_memset memset
- #define gst_tensor_pad_caps_is_static(p) (gst_tensor_pad_get_format (p) == _NNS_TENSOR_FORMAT_STATIC)

  Macro to check current pad caps is static tensor.
- #define gst_tensor_pad_caps_is_flexible(p) (gst_tensor_pad_get_format (p) == _NNS_TENSOR_FORMAT_FLEXIBLE)

  Macro to check current pad caps is flexible tensor.
- #define gst_tensor_pad_caps_is_sparse(p) (gst_tensor_pad_get_format (p) == _NNS_TENSOR_FORMAT_SPARSE)

  Macro to check current pad caps is sparse tensor.
- #define silent_debug(self, ...)

  Macro for debug message.
- #define silent_debug_caps(self, caps, msg)

  Macro for capability debug message.

Typedefs

- typedef struct _tensor_sync_basepad_data tensor_sync_basepad_data

  Tensor Merge/Mux sync data for baspad mode.
- typedef struct _tensor_time_sync_data tensor_time_sync_data

  Tensor Merge/Mux time sync data.

Enumerations

- enum tensor_time_sync_mode {
  SYNC_NOSYNC = 0, SYNC_SLOWEST = 1, SYNC_BASEPAD = 2, SYNC_REFRESH = 3,
  SYNC_END }

  time synchronization options

Functions

- void gst_tensor_parse_config_file (const gchar *config_path, const GObject *object)

  Parses a configuration file and sets the corresponding properties on a GObject.
- tensor_time_sync_mode gst_tensor_time_sync_get_mode (const gchar *str)

  Get the corresponding mode from the string value.
- const gchar * gst_tensor_time_sync_get_mode_string (tensor_time_sync_mode mode)

  Get the time-sync mode string.
- gboolean gst_tensor_time_sync_set_option_data (tensor_time_sync_data *sync)

  Setup time sync option.
- gboolean gst_tensor_time_sync_get_current_time (GstCollectPads *collect, tensor_time_sync_data *sync, GstClockTime *current_time, GstBuffer *tensors_buf)

  A function call to decide current timestamp among collected pads based on PTS. It will decide current timestamp according to sync option. GstMeta is also copied with same sync mode.
- void gst_tensor_time_sync_flush (GstCollectPads *collect)

  A function to be called while processing a flushing event. It should clear old buffer and reset pad data.
- gboolean gst_tensor_time_sync_buffer_from_collectpad (GstCollectPads *collect, tensor_time_sync_data *sync, GstClockTime current_time, GstBuffer *tensors_buf, GstTensorsConfig *configs, gboolean *is_eos)

  A function call to make tensors from collected pads It decide which buffer is going to be used according to sync option.
- GstBuffer * gst_tensor_buffer_from_config (GstBuffer *in, GstTensorsConfig *config)

  Configure gst-buffer with tensors information. NNStreamer handles single memory chunk as single tensor. If incoming buffer has invalid memories, separate it and generate new gst-buffer using tensors information. Note that this function always takes the ownership of input buffer.
- GstCaps * gst_tensor_pad_caps_from_config (GstPad *pad, const GstTensorsConfig *config)

  Get pad caps from tensors config and caps of the peer connected to the pad.
- GstCaps * gst_tensor_pad_possible_caps_from_config (GstPad *pad, const GstTensorsConfig *config)

  Get all possible caps from tensors config. Unlike gst_tensor_pad_caps_from_config(), this function does not check peer caps.
- tensor_format gst_tensor_pad_get_format (GstPad *pad)

  Get tensor format of current pad caps.
- GHashTable * gst_tensor_aggregation_init (void)

  Gets new hash table for tensor aggregation.
- void gst_tensor_aggregation_clear (GHashTable *table, const guint32 key)

  Clears buffers from adapter.
- void gst_tensor_aggregation_clear_all (GHashTable *table)

  Clears buffers from all adapters in hash table.
- GstAdapter * gst_tensor_aggregation_get_adapter (GHashTable *table, const guint32 key)

  Gets adapter from hash table.

9.101.1 Detailed Description

Common header file for NNStreamer, the GStreamer plugin for neural networks.

Date

23 May 2018

See also

https://github.com/nnstreamer/nnstreamer

Author

MyungJoo Ham myungjoo.ham@samsung.com

Bug No known bugs except for NYI items

9.101.2 Macro Definition Documentation

9.101.2.1 gst_tensor_pad_caps_is_flexible

```
#define gst_tensor_pad_caps_is_flexible(
              p ) (gst_tensor_pad_get_format (p) == _NNS_TENSOR_FORMAT_FLEXIBLE)
```

Macro to check current pad caps is flexible tensor.

Definition at line 231 of file tensor_common.h.

9.101.2.2 gst_tensor_pad_caps_is_sparse

```
#define gst_tensor_pad_caps_is_sparse(
              p ) (gst_tensor_pad_get_format (p) == _NNS_TENSOR_FORMAT_SPARSE)
```

Macro to check current pad caps is sparse tensor.

Definition at line 236 of file tensor_common.h.

9.101.2.3 gst_tensor_pad_caps_is_static

```
#define gst_tensor_pad_caps_is_static(
              p ) (gst_tensor_pad_get_format (p) == _NNS_TENSOR_FORMAT_STATIC)
```

Macro to check current pad caps is static tensor.

Definition at line 226 of file tensor_common.h.

9.101.2.4 nns_memcpy

```
#define nns_memcpy memcpy
```

Definition at line 52 of file tensor_common.h.

9.101.2.5 nns_memset

```
#define nns_memset memset
```

Definition at line 53 of file tensor_common.h.

9.101.2.6 silent_debug

```
#define silent_debug(
              self,
              ...  )
```

Value:

```
    do { \
    if (DBG) { \
      GST_DEBUG_OBJECT (self, __VA_ARGS__); \
    } \
  } while (0)
```

Macro for debug message.

Definition at line 276 of file tensor_common.h.

9.101.2.7 silent_debug_caps

```
#define silent_debug_caps(
              self,
              caps,
              msg )
```

Value:

```
  do { \
  if (DBG) { \
    if (caps) { \
      gchar *caps_s_string = gst_caps_to_string (caps); \
      GST_DEBUG_OBJECT (self, msg " = %s\n", caps_s_string); \
      g_free (caps_s_string); \
    } \
  } \
} while (0)
```

Macro for capability debug message.

Definition at line 285 of file tensor_common.h.

9.101.3 Typedef Documentation

9.101.3.1 tensor_sync_basepad_data

```
typedef struct _tensor_sync_basepad_data tensor_sync_basepad_data
```

Tensor Merge/Mux sync data for baspad mode.

9.101.3.2 tensor_time_sync_data

```
typedef struct _tensor_time_sync_data tensor_time_sync_data
```

Tensor Merge/Mux time sync data.

9.101.4 Enumeration Type Documentation

9.101.4.1 tensor_time_sync_mode

```
enum tensor_time_sync_mode
```

time synchronization options

See also

https://github.com/nnstreamer/nnstreamer/wiki/Synchronization-Policies-at-↩Mux-and-Merge

Enumerator

SYNC_NOSYNC	
SYNC_SLOWEST	
SYNC_BASEPAD	
SYNC_REFRESH	
SYNC_END	

Definition at line 62 of file tensor_common.h.

9.101.5 Function Documentation

9.101.5.1 gst_tensor_aggregation_clear()

```
void gst_tensor_aggregation_clear (
            GHashTable * table,
            const guint32 key )
```

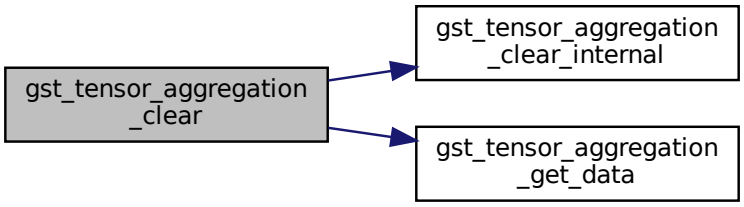
Clears buffers from adapter.

Parameters

table	a hash table instance initialized with gst_tensor_aggregation_init()
key	the key to look up (set null to get default adapter)

Definition at line 763 of file nnstreamer_plugin_api_impl.c.

Here is the call graph for this function:



9.101.5.2 gst_tensor_aggregation_clear_all()

```
void gst_tensor_aggregation_clear_all (
            GHashTable * table )
```

Clears buffers from all adapters in hash table.

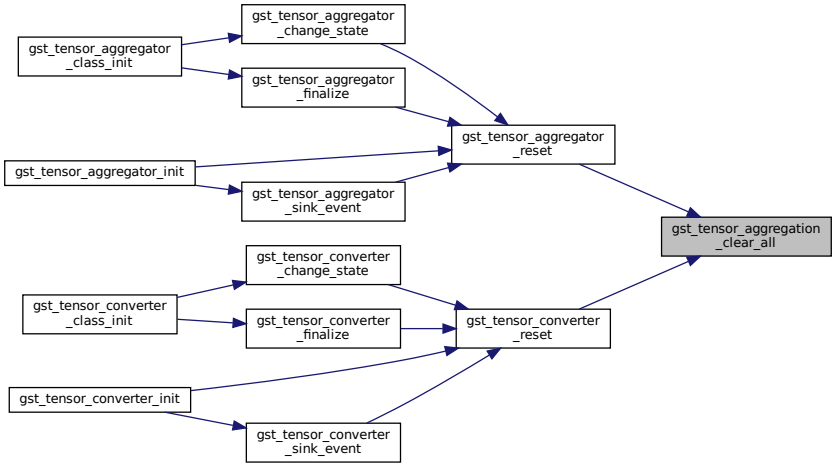
Parameters

table	a hash table instance initialized with gst_tensor_aggregation_init()

Definition at line 778 of file nnstreamer_plugin_api_impl.c.

Here is the call graph for this function:

Here is the caller graph for this function:



9.101.5.3 gst_tensor_aggregation_get_adapter()

```
GstAdapter* gst_tensor_aggregation_get_adapter (
            GHashTable * table,
            const guint32 key )
```

Gets adapter from hash table.

Parameters

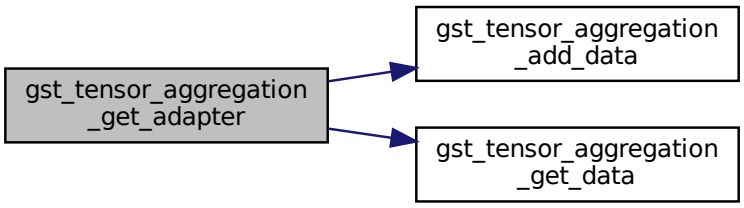
table	a hash table instance initialized with gst_tensor_aggregation_init()
key	the key to look up (set null to get default adapter)

Returns

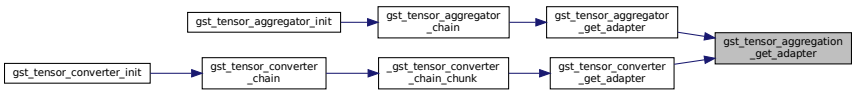
gst-adapter instance. DO NOT release this instance.

Definition at line 790 of file nnstreamer_plugin_api_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.101.5.4 gst_tensor_aggregation_init()

```
GHashTable* gst_tensor_aggregation_init (
            void  )
```

Gets new hash table for tensor aggregation.

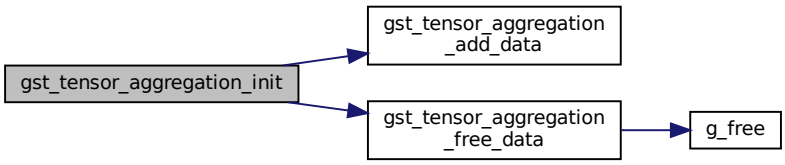
Returns

Newly allocated hash table, caller should release this using g_hash_table_destroy().

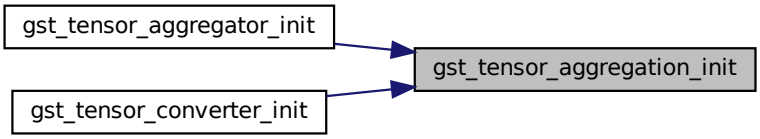
Add default adapter (for the case if buffer has no specific id). If gst-buffer has tensor-meta which includes client-id, e.g., aggregation frames from multiple clients on query-server pipeline, nnstreamer element should parse meta and request adapter with this id. However, on normal pipeline, gst-buffer does not contain tensor-meta, then the element may request adapter with null key string.

Definition at line 737 of file nnstreamer_plugin_api_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.101.5.5 gst_tensor_buffer_from_config()

```
GstBuffer* gst_tensor_buffer_from_config (
            GstBuffer * in,
            GstTensorsConfig * config )
```

Configure gst-buffer with tensors information. NNStreamer handles single memory chunk as single tensor. If incoming buffer has invalid memories, separate it and generate new gst-buffer using tensors information. Note that this function always takes the ownership of input buffer.

Parameters

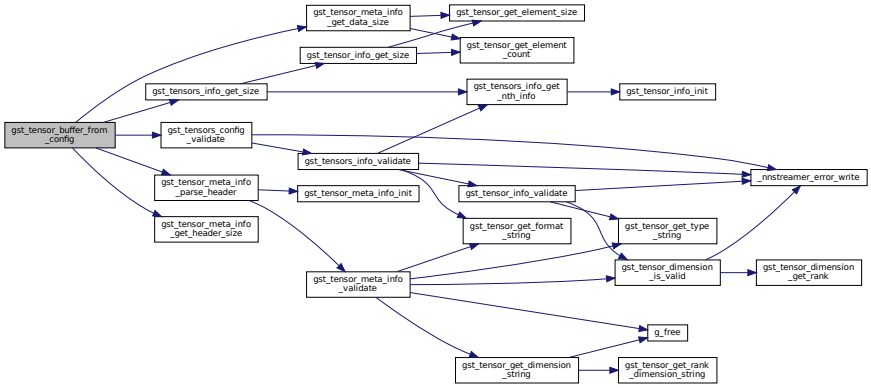
in	input buffer
config	tensors config structure

Returns

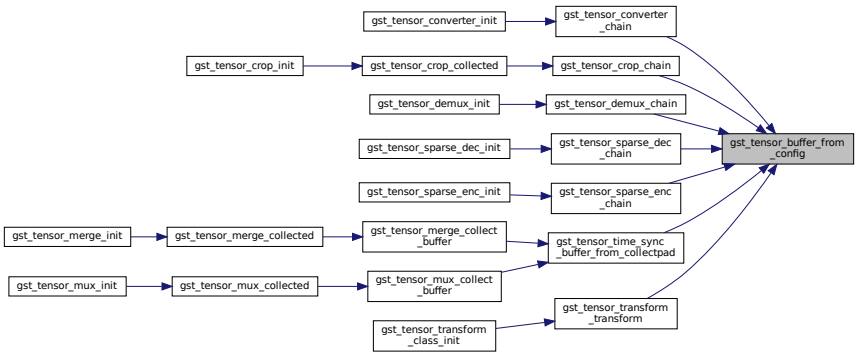
Newly allocated buffer. Null if failed. Caller should unref the buffer using gst_buffer_unref().

Definition at line 535 of file nnstreamer_plugin_api_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.101.5.6 gst_tensor_pad_caps_from_config()

```
GstCaps* gst_tensor_pad_caps_from_config (
            GstPad * pad,
            const GstTensorsConfig * config )
```

Get pad caps from tensors config and caps of the peer connected to the pad.

Parameters

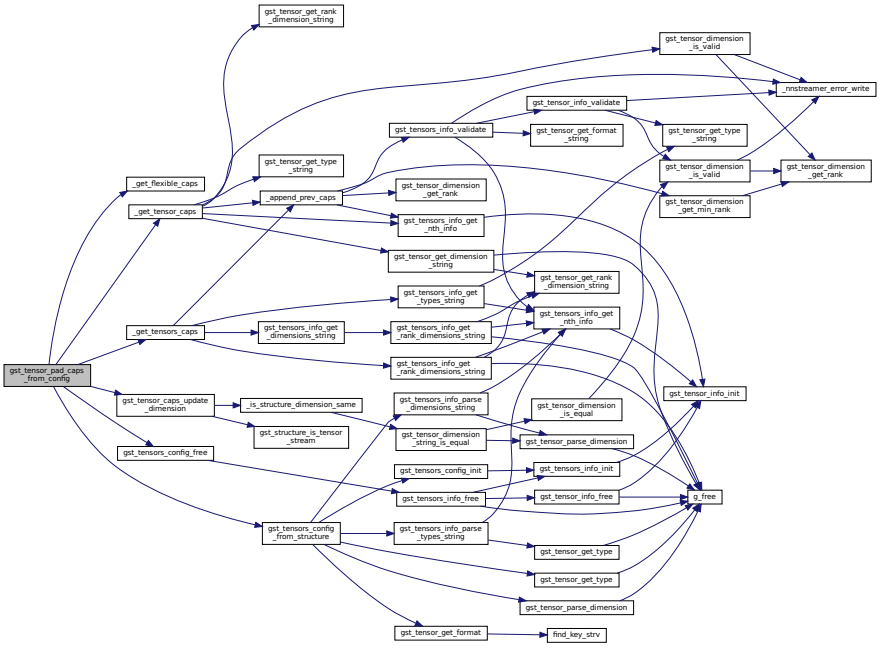
pad	GstPad to get possible caps
config	tensors config structure

Returns

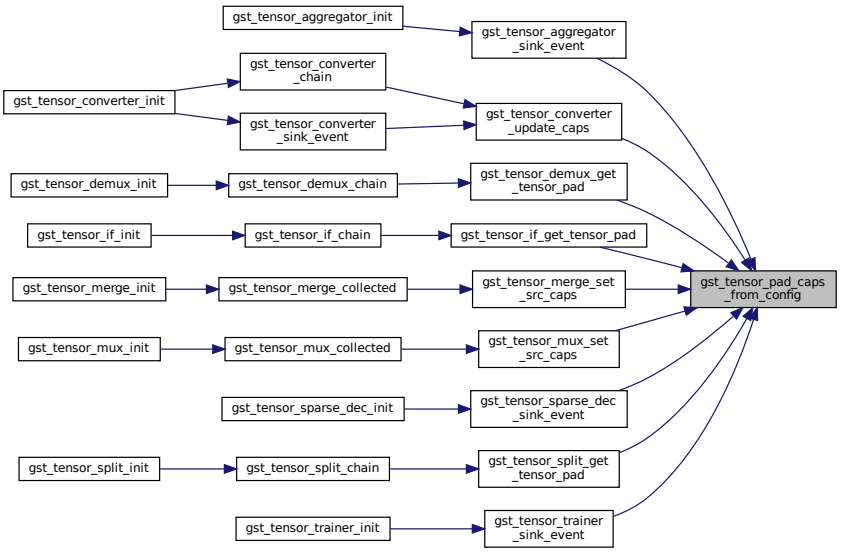
caps for given config. Caller is responsible for unreffing the returned caps.

Definition at line 1209 of file nnstreamer_plugin_api_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.101.5.7 gst_tensor_pad_get_format()

```
tensor_format gst_tensor_pad_get_format (
            GstPad * pad )
```

Get tensor format of current pad caps.

Parameters

pad	GstPad to check current caps.

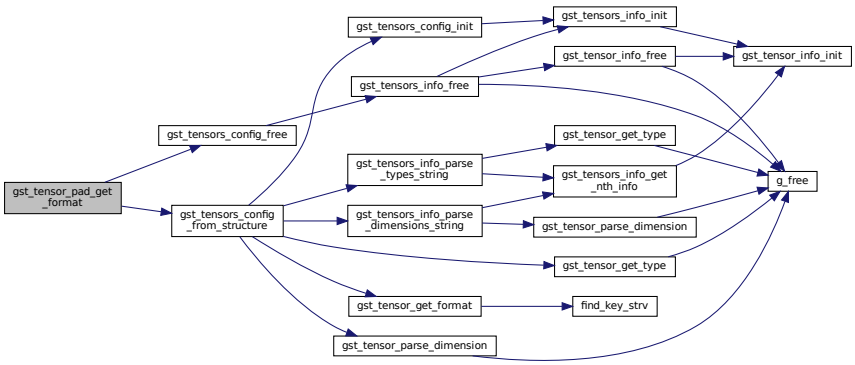
Returns

The tensor_format of current pad caps.

If pad does not have tensor caps return _NNS_TENSOR_FORMAT_END

Definition at line 1343 of file nnstreamer_plugin_api_impl.c.

Here is the call graph for this function:



9.101.5.8 gst_tensor_pad_possible_caps_from_config()

```
GstCaps* gst_tensor_pad_possible_caps_from_config (
            GstPad * pad,
            const GstTensorsConfig * config )
```

Get all possible caps from tensors config. Unlike gst_tensor_pad_caps_from_config(), this function does not check peer caps.

Parameters

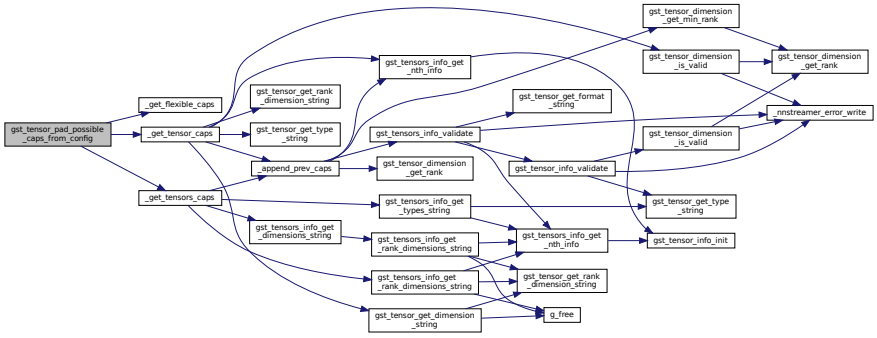
pad	GstPad to get possible caps
config	tensors config structure

Returns

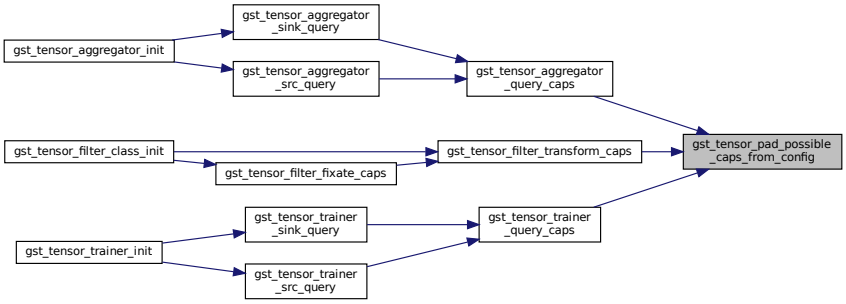
caps for given config. Caller is responsible for unreffing the returned caps.

Definition at line 1286 of file nnstreamer_plugin_api_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.101.5.9 gst_tensor_parse_config_file()

```
void gst_tensor_parse_config_file (
            const gchar * config_path,
            const GObject * object )
```

Parses a configuration file and sets the corresponding properties on a GObject.

This function reads the contents of the configuration file located at the given path and sets the properties of the specified GObject based on the configuration data.

Parameters

config_path	The path to the configuration file.
object	The GObject on which to set the properties.

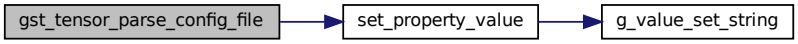
Note

The responsibility of managing the memory of the GObject passed as a parameter lies outside this function.

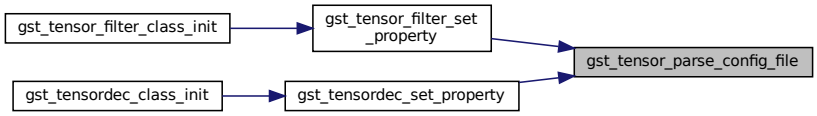
Iterate over each line

Definition at line 1902 of file nnstreamer_plugin_api_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.101.5.10 gst_tensor_time_sync_buffer_from_collectpad()

```
gboolean gst_tensor_time_sync_buffer_from_collectpad (
            GstCollectPads * collect,
            tensor_time_sync_data * sync,
            GstClockTime current_time,
            GstBuffer * tensors_buf,
            GstTensorsConfig * configs,
            gboolean * is_eos )
```

A function call to make tensors from collected pads It decide which buffer is going to be used according to sync option.

Returns

True to push buffer.

Parameters

collect	Collect pad.
sync	Synchronization Option (NOSYNC, SLOWEST, BASEPAD, END)
current_time	Current Timestamp
tensors_buf	Generated GstBuffer for Collected Buffer
configs	Configuration Info for Collected Buffer
is_eos	True when EOS (end-of-stream)

A function call to make tensors from collected pads It decide which buffer is going to be used according to sync option.

Returns

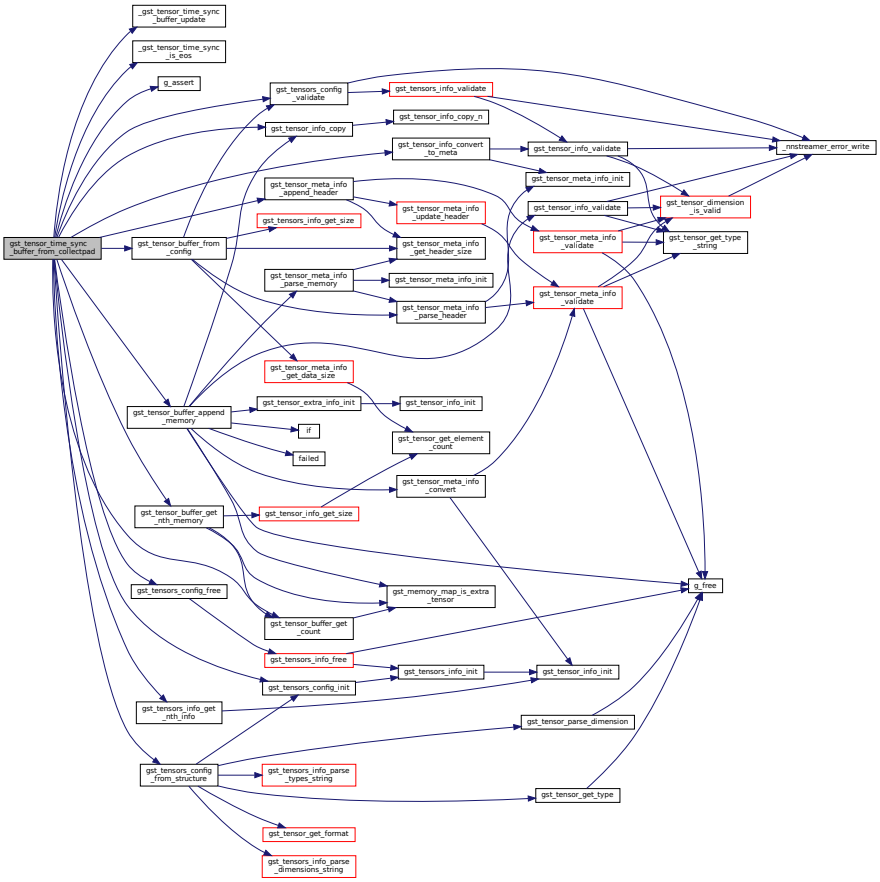
True to push buffer.

This would be an internal logic error. in_configs should be already confirmed valid at the negotiation phase and this function should be called in a running pipeline. If new sync mode is enabled (e.g., handle output when a pad gets new buffer), this may cause unexpected exception.

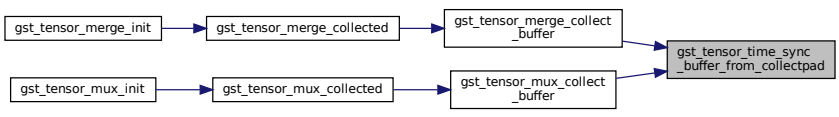
These are internal logic error. If given inputs are incorrect, the negotiation should have been failed before this stage.

Definition at line 332 of file nnstreamer_plugin_api_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.101.5.11 gst_tensor_time_sync_flush()

```
void gst_tensor_time_sync_flush (
            GstCollectPads * collect )
```

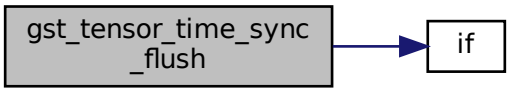
A function to be called while processing a flushing event. It should clear old buffer and reset pad data.

Parameters

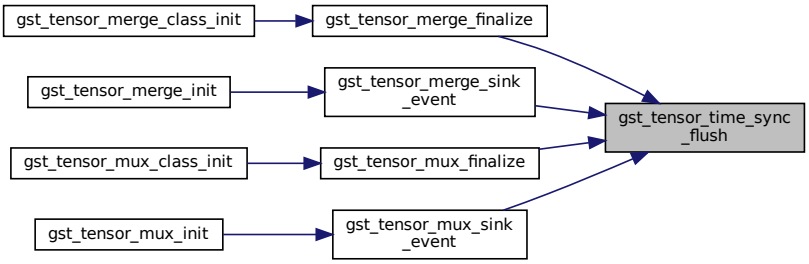
collect	Collect pad.

Definition at line 263 of file nnstreamer_plugin_api_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.101.5.12 gst_tensor_time_sync_get_current_time()

```
gboolean gst_tensor_time_sync_get_current_time (
            GstCollectPads * collect,
            tensor_time_sync_data * sync,
            GstClockTime * current_time,
            GstBuffer * tensors_buf )
```

A function call to decide current timestamp among collected pads based on PTS. It will decide current timestamp according to sync option. GstMeta is also copied with same sync mode.

Returns

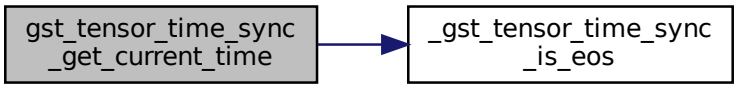
True / False. If EOS, it return TRUE.

Parameters

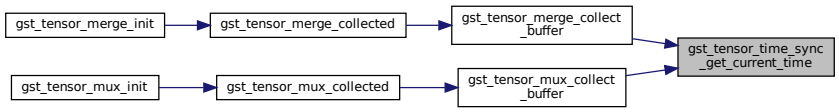
collect	Collect pad.
sync	Synchronization Option (NOSYNC, SLOWEST, BASEPAD, END)
current_time	Current time
tensors_buf	Generated GstBuffer for Collected Buffer

Definition at line 203 of file nnstreamer_plugin_api_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.101.5.13 gst_tensor_time_sync_get_mode()

```
tensor_time_sync_mode gst_tensor_time_sync_get_mode (
            const gchar * str )
```

Get the corresponding mode from the string value.

Parameters

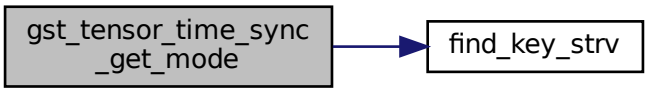
in	*str*	The string value for the mode.

Returns

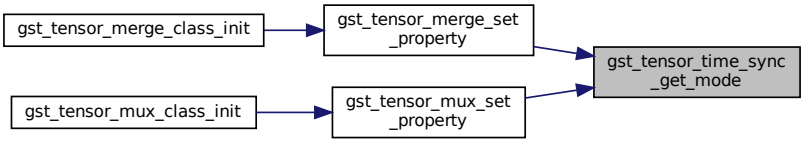
Corresponding mode for the string. SYNC_END for errors.

Definition at line 101 of file nnstreamer_plugin_api_impl.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.101.5.14 gst_tensor_time_sync_get_mode_string()

```
const gchar* gst_tensor_time_sync_get_mode_string (
            tensor_time_sync_mode mode )
```

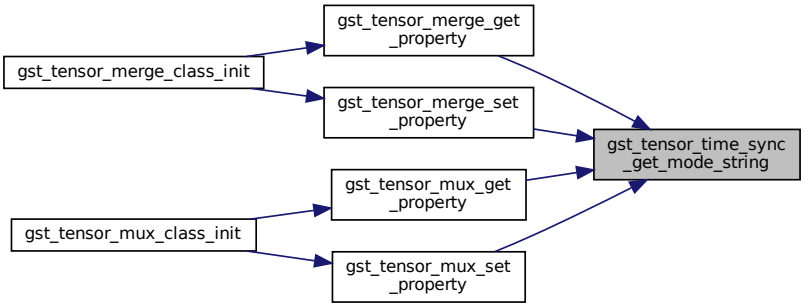
Get the time-sync mode string.

Returns

Corresponding mode string.

Definition at line 115 of file nnstreamer_plugin_api_impl.c.

Here is the caller graph for this function:



9.101.5.15 gst_tensor_time_sync_set_option_data()

```
gboolean gst_tensor_time_sync_set_option_data (
            tensor_time_sync_data * sync )
```

Setup time sync option.

Parameters

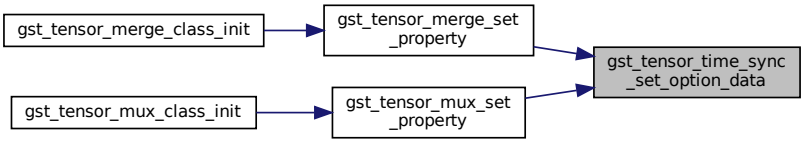
[in/out]	filter "this" pointer. Sync mode & option MUST BE set already.

Returns

True if successfully set the option.

Definition at line 126 of file nnstreamer_plugin_api_impl.c.

Here is the caller graph for this function:

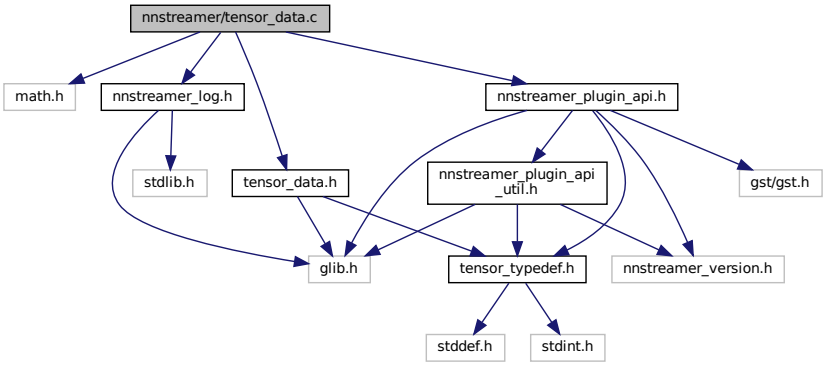


9.102 nnstreamer/tensor_data.c File Reference

Internal functions to handle various tensor type and value.

```
#include <math.h>
#include "tensor_data.h"
#include "nnstreamer_log.h"
#include "nnstreamer_plugin_api.h"
```

Include dependency graph for tensor_data.c:



Macros

- #define td_set_data(td, v, dtype)

  Macro to set data in struct.
- #define td_get_data(td, v, dtype)
- #define td_typecast_to(td, itype, otype)
- #define td_typecast_to_fromf16(td, itype)
- #define td_typecast(td, otype)

Functions

- while (0)

  *dtype = *((dtype *) v); *
- gboolean gst_tensor_data_get (tensor_data_s *td, gpointer value)

  Get tensor element value.
- gboolean gst_tensor_data_typecast (tensor_data_s *td, tensor_type type)

  Typecast tensor element data.
- gboolean gst_tensor_data_raw_typecast (gpointer input, tensor_type in_type, gpointer output, tensor_type out_type)

  Typecast tensor element value.
- gboolean gst_tensor_data_raw_average (gpointer raw, gsize length, tensor_type type, gdouble **result)

  Calculate average value of the tensor.
- gboolean gst_tensor_data_raw_average_per_channel (gpointer raw, gsize length, tensor_type type, tensor_dim dim, gdouble **results)

  Calculate average value of the tensor per channel (the first dim).
- gboolean gst_tensor_data_raw_std (gpointer raw, gsize length, tensor_type type, gdouble *average, gdouble **result)

  Calculate standard deviation of the tensor.
- gboolean gst_tensor_data_raw_std_per_channel (gpointer raw, gsize length, tensor_type type, tensor_dim dim, gdouble *averages, gdouble **results)

  Calculate standard deviation of the tensor per channel (the first dim).

9.102.1 Detailed Description

Internal functions to handle various tensor type and value.

Date

10 Mar 2021

See also

http://github.com/nnstreamer/nnstreamer

Author

Jaeyun Jung jy1210.jung@samsung.com

Bug No known bugs except for NYI items

9.102.2 Macro Definition Documentation

9.102.2.1 td_get_data

```
#define td_get_data(
                td,
                v,
                dtype )
```

Value:

```
do { \
*((dtype *) v) = (td)->data._
```

9.102.2.2 td_set_data

```
#define td_set_data(
                td,
                v,
                dtype )
```

Value:

```
do { \
(td)->data._
```

Macro to set data in struct.

Definition at line 21 of file tensor_data.c.

9.102.2.3 td_typecast

```
#define td_typecast(
              td,
              otype )
```

Value:

```
    do { \
    switch ((td)->type) { \
      case _NNS_INT32: td_typecast_to (td, int32_t, otype); break; \
      case _NNS_UINT32: td_typecast_to (td, uint32_t, otype); break; \
      case _NNS_INT16: td_typecast_to (td, int16_t, otype); break; \
      case _NNS_UINT16:  td_typecast_to (td, uint16_t, otype); break; \
      case _NNS_INT8: td_typecast_to (td, int8_t, otype); break; \
      case _NNS_UINT8: td_typecast_to (td, uint8_t, otype); break; \
      case _NNS_FLOAT64: td_typecast_to (td, double, otype); break; \
      case _NNS_FLOAT32: td_typecast_to (td, float, otype); break; \
      case _NNS_FLOAT16: td_typecast_to_fromf16 (td, otype); break; \
      case _NNS_INT64: td_typecast_to (td, int64_t, otype); break; \
      case _NNS_UINT64: td_typecast_to (td, uint64_t, otype); break; \
      default: g_assert (0); break; \
    } \
  } while (0)
```

9.102.2.4 td_typecast_to

```
#define td_typecast_to(
              td,
              itype,
              otype )
```

Value:

```
    do { \
    itype in_val = (td)->data._
```

9.102.2.5 td_typecast_to_fromf16

```
#define td_typecast_to_fromf16(
              td,
              itype )
```

Value:

```
    do { \
    nns_loge ("Your nnstreamer binary is built without -DFLOAT16_SUPPORT option; thus float16 is not
       supported.\n"); \
    g_assert (0); \
  } while (0)
```

9.102.3 Function Documentation

9.102.3.1 gst_tensor_data_get()

```
gboolean gst_tensor_data_get (
            tensor_data_s * td,
            gpointer value )
```

Get tensor element value.

Parameters

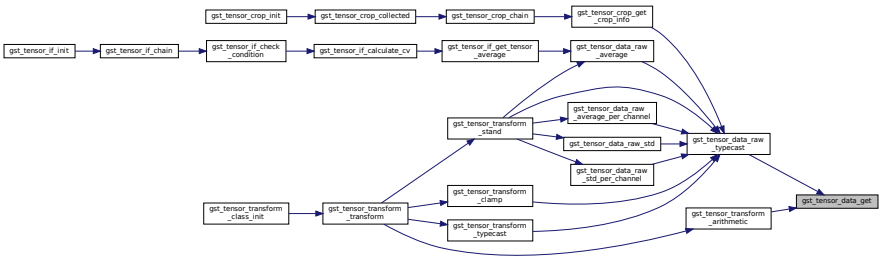
td	struct for tensor data
value	pointer of tensor element value

Returns

TRUE if no error

Definition at line 143 of file tensor_data.c.

Here is the caller graph for this function:



9.102.3.2 gst_tensor_data_raw_average()

```
gboolean gst_tensor_data_raw_average (
            gpointer raw,
            gsize length,
            tensor_type type,
            gdouble ** result )
```

Calculate average value of the tensor.

Parameters

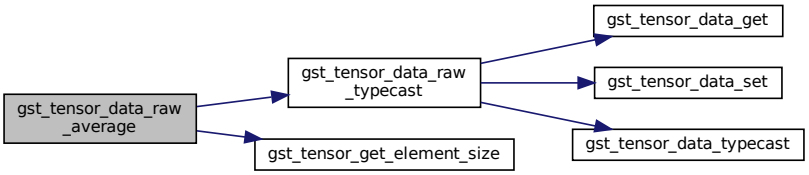
raw	pointer of raw tensor data
length	byte size of raw tensor data
type	tensor type
result	double pointer for average value of given tensor. Caller should release allocated memory.

Returns

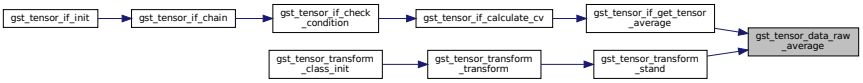
TRUE if no error

Definition at line 315 of file tensor_data.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.102.3.3 gst_tensor_data_raw_average_per_channel()

```
gboolean gst_tensor_data_raw_average_per_channel (
            gpointer raw,
            gsize length,
            tensor_type type,
            tensor_dim dim,
            gdouble ** results )
```

Calculate average value of the tensor per channel (the first dim).

Parameters

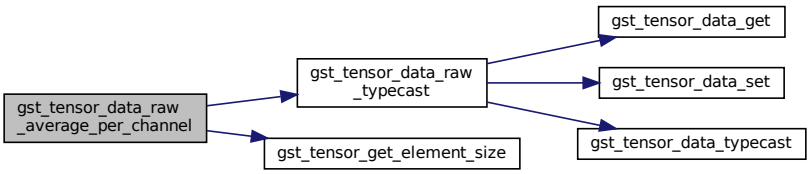
raw	pointer of raw tensor data
length	byte size of raw tensor data
type	tensor type
tensor_dim	tensor dimension
results	double array contains average values of each channel. Caller should release allocated array.

Returns

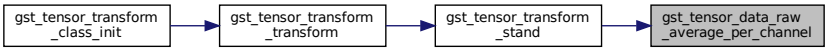
TRUE if no error

Definition at line 360 of file tensor_data.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.102.3.4 gst_tensor_data_raw_std()

```
gboolean gst_tensor_data_raw_std (
            gpointer raw,
            gsize length,
            tensor_type type,
            gdouble * average,
            gdouble ** result )
```

Calculate standard deviation of the tensor.

Parameters

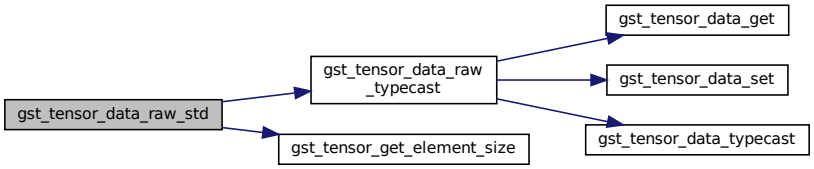
raw	pointer of raw tensor data
length	byte size of raw tensor data
type	tensor type
average	average value of given tensor
result	double pointer for standard deviation of given tensor. Caller should release allocated memory.

Returns

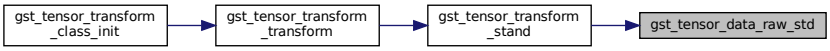
TRUE if no error

Definition at line 409 of file tensor_data.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.102.3.5 gst_tensor_data_raw_std_per_channel()

```
gboolean gst_tensor_data_raw_std_per_channel (
            gpointer raw,
            gsize length,
            tensor_type type,
            tensor_dim dim,
            gdouble * averages,
            gdouble ** results )
```

Calculate standard deviation of the tensor per channel (the first dim).

Parameters

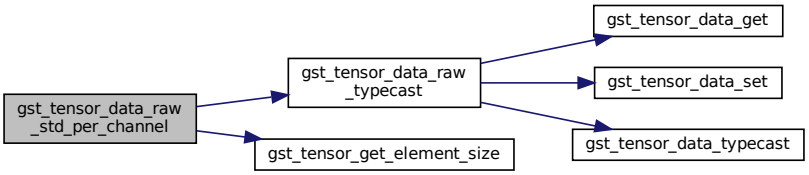
raw	pointer of raw tensor data
length	byte size of raw tensor data
type	tensor type
tensor_dim	tensor dimension
averages	average values of given tensor per-channel
results	double array contains standard deviation of each channel. Caller should release allocated array.

Returns

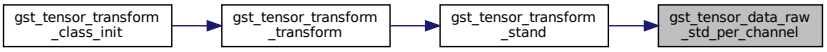
TRUE if no error

Definition at line 455 of file tensor_data.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.102.3.6 gst_tensor_data_raw_typecast()

```
gboolean gst_tensor_data_raw_typecast (
            gpointer input,
            tensor_type in_type,
            gpointer output,
            tensor_type out_type )
```

Typecast tensor element value.

Parameters

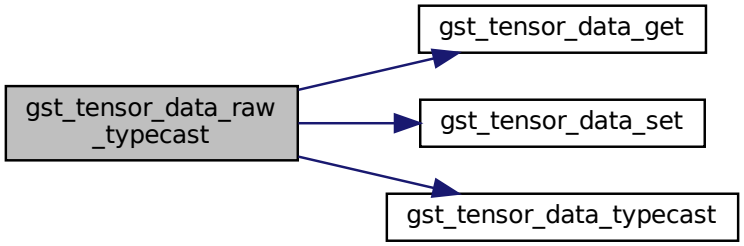
input	pointer of input tensor data
in_type	input tensor type
output	pointer of output tensor data
out_type	output tensor type

Returns

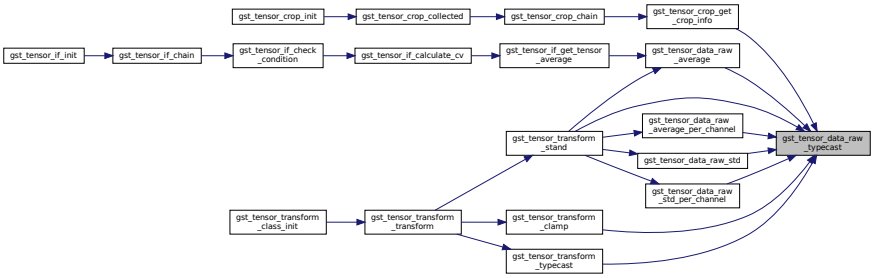
TRUE if no error

Definition at line 290 of file tensor_data.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.102.3.7 gst_tensor_data_typecast()

```
gboolean gst_tensor_data_typecast (
            tensor_data_s * td,
            tensor_type type )
```

Typecast tensor element data.

Parameters

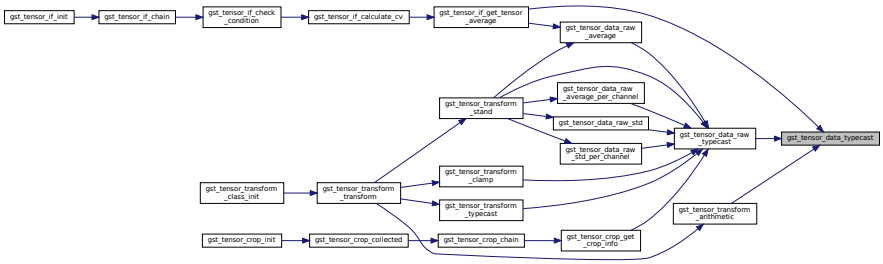
td	struct for tensor data
type	tensor type to be transformed

Returns

TRUE if no error

Definition at line 203 of file tensor_data.c.

Here is the caller graph for this function:



9.102.3.8 while()

```
while (
              0  )
```

dtype = ∗((dtype ∗) v); \

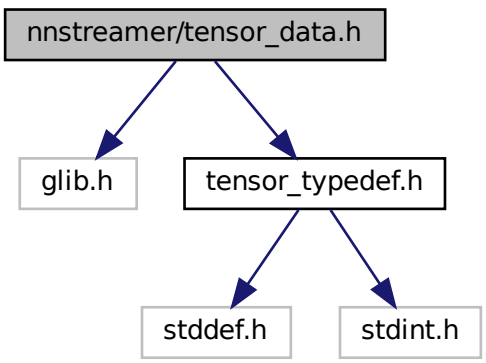
Definition at line 23 of file tensor_data.c.

9.103 nnstreamer/tensor_data.h File Reference

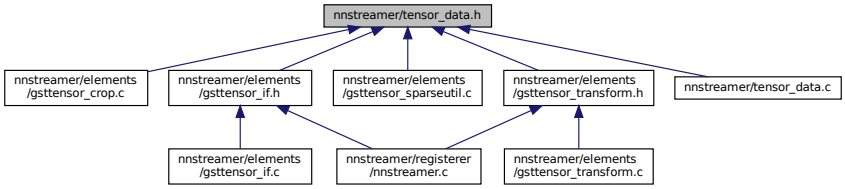
Internal functions to handle various tensor type and value.

```
#include <glib.h>
#include <tensor_typedef.h>
```

Include dependency graph for tensor_data.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct tensor_data_s

  Structure for tensor data.

Functions

- gboolean gst_tensor_data_set (tensor_data_s *td, tensor_type type, gpointer value)

  Set tensor element data with given type.
- gboolean gst_tensor_data_get (tensor_data_s *td, gpointer value)

  Get tensor element value.
- gboolean gst_tensor_data_typecast (tensor_data_s *td, tensor_type type)

  Typecast tensor element data.
- gboolean gst_tensor_data_raw_typecast (gpointer input, tensor_type in_type, gpointer output, tensor_type out_type)

  Typecast tensor element value.
- gboolean gst_tensor_data_raw_average (gpointer raw, gsize length, tensor_type type, gdouble **result)

  Calculate average value of the tensor.
- gboolean gst_tensor_data_raw_average_per_channel (gpointer raw, gsize length, tensor_type type, tensor_dim dim, gdouble **results)

  Calculate average value of the tensor per channel (the first dim).
- gboolean gst_tensor_data_raw_std (gpointer raw, gsize length, tensor_type type, gdouble *average, gdouble **result)

  Calculate standard deviation of the tensor.
- gboolean gst_tensor_data_raw_std_per_channel (gpointer raw, gsize length, tensor_type type, tensor_dim dim, gdouble *averages, gdouble **results)

  Calculate standard deviation of the tensor per channel (the first dim).

9.103.1 Detailed Description

Internal functions to handle various tensor type and value.

Date

10 Mar 2021

See also

http://github.com/nnstreamer/nnstreamer

Author

Jaeyun Jung jy1210.jung@samsung.com

Bug No known bugs except for NYI items

9.103.2 Function Documentation

9.103.2.1 gst_tensor_data_get()

```
gboolean gst_tensor_data_get (
            tensor_data_s * td,
            gpointer value )
```

Get tensor element value.

Parameters

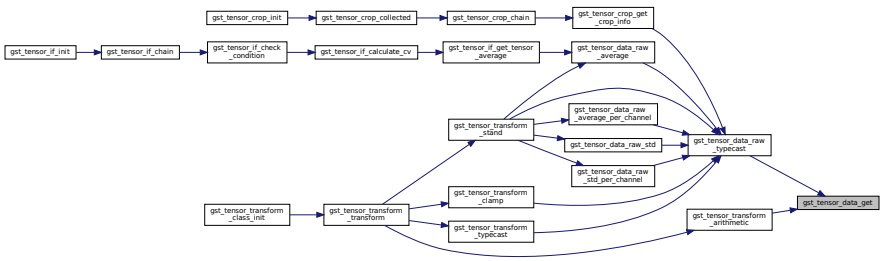
td	struct for tensor data
value	pointer of tensor element value

Returns

TRUE if no error

Definition at line 143 of file tensor_data.c.

Here is the caller graph for this function:



9.103.2.2 gst_tensor_data_raw_average()

```
gboolean gst_tensor_data_raw_average (
            gpointer raw,
            gsize length,
            tensor_type type,
            gdouble ** result )
```

Calculate average value of the tensor.

Parameters

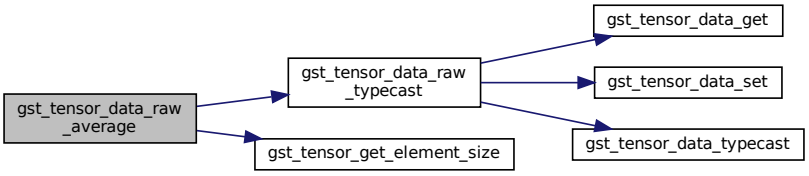
raw	pointer of raw tensor data
length	byte size of raw tensor data
type	tensor type
result	double pointer for average value of given tensor. Caller should release allocated memory.

Returns

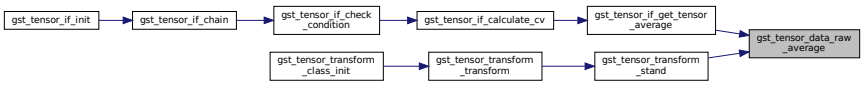
TRUE if no error

Definition at line 315 of file tensor_data.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.103.2.3 gst_tensor_data_raw_average_per_channel()

```
gboolean gst_tensor_data_raw_average_per_channel (
            gpointer raw,
            gsize length,
            tensor_type type,
            tensor_dim dim,
            gdouble ** results )
```

Calculate average value of the tensor per channel (the first dim).

Parameters

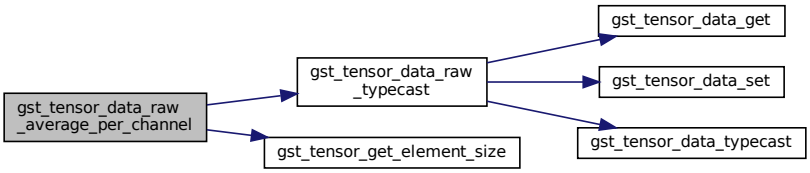
raw	pointer of raw tensor data
length	byte size of raw tensor data
type	tensor type
tensor_dim	tensor dimension
results	double array contains average values of each channel. Caller should release allocated array.

Returns

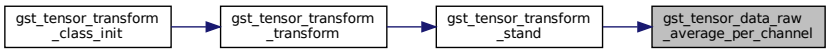
TRUE if no error

Definition at line 360 of file tensor_data.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.103.2.4 gst_tensor_data_raw_std()

```
gboolean gst_tensor_data_raw_std (
            gpointer raw,
            gsize length,
            tensor_type type,
            gdouble * average,
            gdouble ** result )
```

Calculate standard deviation of the tensor.

Parameters

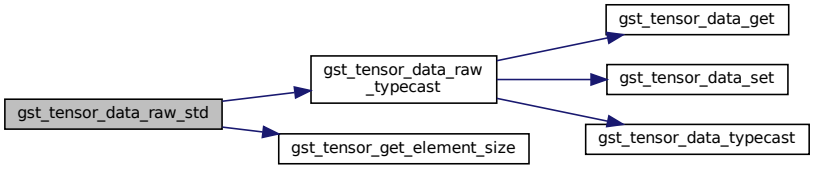
raw	pointer of raw tensor data
length	byte size of raw tensor data
type	tensor type
average	average value of given tensor
result	double pointer for standard deviation of given tensor. Caller should release allocated memory.

Returns

TRUE if no error

Definition at line 409 of file tensor_data.c.

Here is the call graph for this function:



Here is the caller graph for this function:

9.103.2.5 gst_tensor_data_raw_std_per_channel()

```
gboolean gst_tensor_data_raw_std_per_channel (
            gpointer raw,
            gsize length,
            tensor_type type,
            tensor_dim dim,
            gdouble * averages,
            gdouble ** results )
```

Calculate standard deviation of the tensor per channel (the first dim).

Parameters

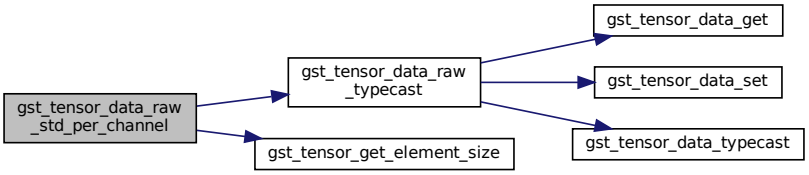
raw	pointer of raw tensor data
length	byte size of raw tensor data
type	tensor type
tensor_dim	tensor dimension
averages	average values of given tensor per-channel
results	double array contains standard deviation of each channel. Caller should release allocated array.

Returns

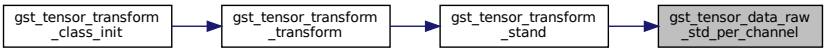
TRUE if no error

Definition at line 455 of file tensor_data.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.103.2.6 gst_tensor_data_raw_typecast()

```
gboolean gst_tensor_data_raw_typecast (
            gpointer input,
            tensor_type in_type,
            gpointer output,
            tensor_type out_type )
```

Typecast tensor element value.

Parameters

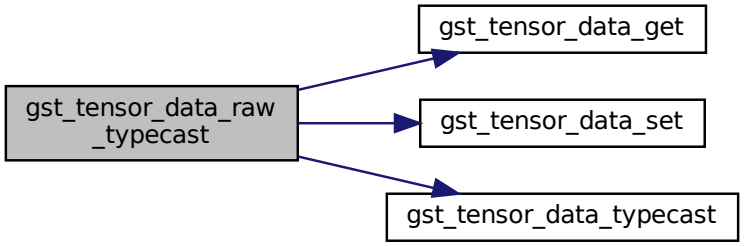
input	pointer of input tensor data
in_type	input tensor type
output	pointer of output tensor data
out_type	output tensor type

Returns

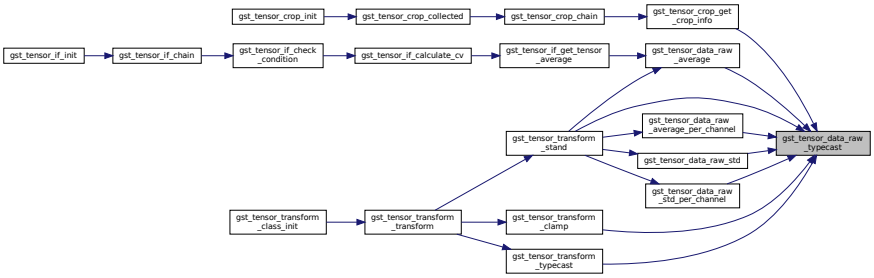
TRUE if no error

Definition at line 290 of file tensor_data.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.103.2.7 gst_tensor_data_set()

```
gboolean gst_tensor_data_set (
            tensor_data_s * td,
            tensor_type type,
            gpointer value )
```

Set tensor element data with given type.

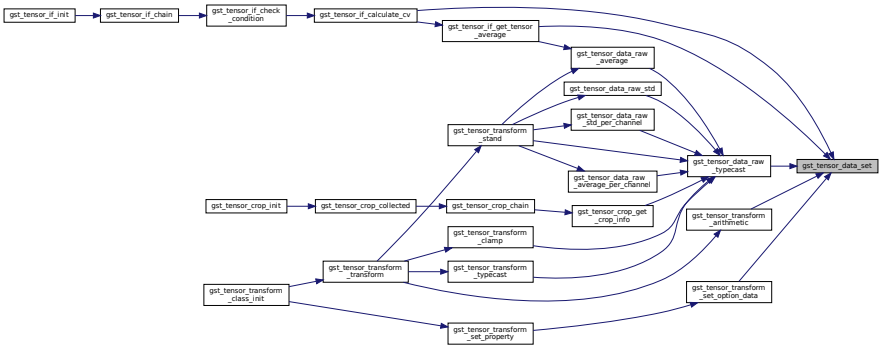
Parameters

td	struct for tensor data
type	tensor type
value	pointer of tensor element value

Returns

TRUE if no error

Here is the caller graph for this function:



9.103.2.8 gst_tensor_data_typecast()

```
gboolean gst_tensor_data_typecast (
            tensor_data_s * td,
            tensor_type type )
```

Typecast tensor element data.

Parameters

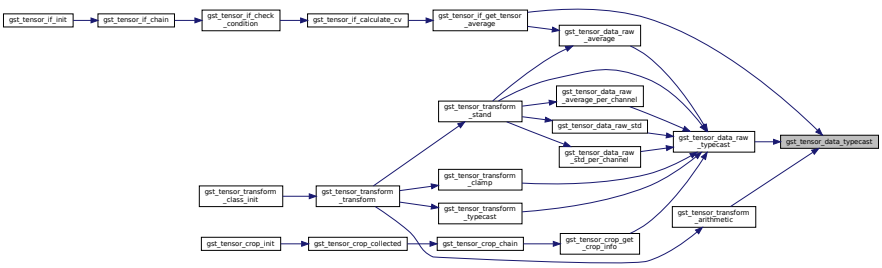
td	struct for tensor data
type	tensor type to be transformed

Returns

TRUE if no error

Definition at line 203 of file tensor_data.c.

Here is the caller graph for this function:

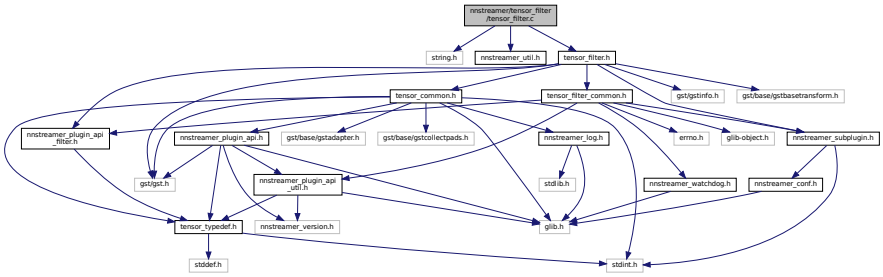


9.104 nnstreamer/tensor_filter/tensor_filter.c File Reference

GStreamer plugin to use general neural network frameworks as filters.

```
#include <string.h>
#include <nnstreamer_util.h>
#include "tensor_filter.h"
```

Include dependency graph for tensor_filter.c:



Classes

- struct _FilterTransformData

  Internal data structure for tensor_filter transform data.

Macros

- #define EVENT_NAME_UPDATE_MODEL "evt_update_model"
- #define GST_CAT_DEFAULT gst_tensor_filter_debug
- #define TF_MODELNAME(prop) ((prop)->model_files ? ((prop)->model_files[0]) : "[No Model File]")
- #define CAPS_STRING GST_TENSOR_CAP_DEFAULT ";" GST_TENSORS_CAP_MAKE ("{ static, flexible }")

  Default caps string for both sink and source pad.
- #define gst_tensor_filter_parent_class parent_class
- #define LATENCY_REPORT_HEADROOM 0.05

  Headroom (extra duration) added to actual latency estimate reported to LATENCY query, to limit number of updates when tracking the maximum value - arbitrarily set to 5%.
- #define LATENCY_REPORT_THRESHOLD 0.25

  Threshold deciding when tracking latency estimate that current value is sufficiently lower than reported value so that a notification update is necessary - arbitrarily set to 25%.
- #define THRESHOLD_DROP_OLD (2000)
- #define THRESHOLD_CACHE_OLD (1000)

Typedefs

- typedef struct _FilterTransformData FilterTransformData

  Internal data structure for tensor_filter transform data.

Functions

- GST_DEBUG_CATEGORY_STATIC (gst_tensor_filter_debug)
- G_DEFINE_TYPE (GstTensorFilter, gst_tensor_filter, GST_TYPE_BASE_TRANSFORM)
- static void gst_tensor_filter_set_property (GObject *object, guint prop_id, const GValue *value, GParamSpec *pspec)

  Setter for tensor_filter properties.
- static void gst_tensor_filter_get_property (GObject *object, guint prop_id, GValue *value, GParamSpec *pspec)

  Getter for tensor_filter properties.
- static void gst_tensor_filter_finalize (GObject *object)

  Function to finalize instance.
- static GstFlowReturn gst_tensor_filter_transform (GstBaseTransform *trans, GstBuffer *inbuf, GstBuffer *outbuf)

  non-ip transform. required vmethod of GstBaseTransform.
- static GstCaps * gst_tensor_filter_transform_caps (GstBaseTransform *trans, GstPadDirection direction, GstCaps *caps, GstCaps *filter)

  configure tensor-srcpad cap from "proposed" cap.
- static GstCaps * gst_tensor_filter_fixate_caps (GstBaseTransform *trans, GstPadDirection direction, GstCaps *caps, GstCaps *othercaps)

  fixate caps. required vmethod of GstBaseTransform.
- static gboolean gst_tensor_filter_set_caps (GstBaseTransform *trans, GstCaps *incaps, GstCaps *outcaps)

  set caps. required vmethod of GstBaseTransform.
- static gboolean gst_tensor_filter_query (GstBaseTransform *trans, GstPadDirection direction, GstQuery *query)

  query handling, optional vmethod of GstBaseTransform.
- static gboolean gst_tensor_filter_transform_size (GstBaseTransform *trans, GstPadDirection direction, GstCaps *caps, gsize size, GstCaps *othercaps, gsize *othersize)

  Tell the framework the required size of buffer based on the info of the other side pad. optional vmethod of BaseTransform.
- static gboolean gst_tensor_filter_start (GstBaseTransform *trans)

  Called when the element starts processing. optional vmethod of BaseTransform.
- static gboolean gst_tensor_filter_stop (GstBaseTransform *trans)

  Called when the element stops processing. optional vmethod of BaseTransform.
- static gboolean gst_tensor_filter_sink_event (GstBaseTransform *trans, GstEvent *event)

  Event handler for sink pad of tensor filter.
- static gboolean gst_tensor_filter_src_event (GstBaseTransform *trans, GstEvent *event)

  Event handler for src pad of tensor filter.
- static void gst_tensor_filter_class_init (GstTensorFilterClass *klass)

  initialize the tensor_filter's class
- static void gst_tensor_filter_init (GstTensorFilter *self)

  initialize the new element instantiate pads and add them to element set pad callback functions initialize instance structure
- static gsize gst_tensor_filter_get_tensor_size (GstTensorFilter *self, guint index, gboolean is_input)

  Calculate tensor buffer size.
- static void gst_tensor_filter_destroy_notify (void *data)

  Free the data allocated for tensor transform.
- static GstMemory * gst_tensor_filter_get_wrapped_mem (GstTensorFilter *self, gpointer data, gsize size)

  Allocate new memory block from given data.
- static void prepare_statistics (GstTensorFilterPrivate *priv)

  Prepare statistics for performance profiling (e.g, latency, throughput)
- static void accumulate_latency (void *data, void *user_data)

Helper function to accumulate latencies.

- static void record_statistics (GstTensorFilterPrivate *priv)

  Record statistics for performance profiling (e.g, latency, throughput)

- static void track_latency (GstTensorFilter *self)

  Track estimated latency and notify pipeline when it changes. Latency estimates may be a bit jittery. On the principle we want to inform pipeline with the latency from longest inference. However, first inference may take much longer, or model filter configuration may change. Therefore any change of more than 10% (arbitrary value) to a lower latency is also reported to pipeline. Notification is done sending LATENCY message to bus. Upon receipt, application will initiate a pipeline latency probe via LATENCY query.

- static gboolean gst_tensor_filter_check_throttling_delay (GstBaseTransform *trans, GstBuffer *inbuf)

  Check throttling delay and send qos overflow event to upstream elements.

- static GstFlowReturn _gst_tensor_filter_transform_validate (GstBaseTransform *trans, GstBuffer *inbuf, GstBuffer *outbuf)

  Check input parameters for gst_tensor_filter_transform ();.

- static void _gst_tensor_filter_release_mem_until_idx (FilterTransformData *trans_data, guint end_index)

  Internal function to release mem and unmap info with index.

- static gsize _gst_tensor_filter_convert_meta (FilterTransformData *trans_data, GstTensorsInfo *info, guint idx)

  Internal function to convert tensor meta and get header size of flexible tensor.

- static FilterTransformData * _gst_tensor_filter_transform_get_all_input_data (GstBaseTransform *trans, GstBuffer *buf)

  Internal function to get input tensors.

- static GstTensorMemory * _gst_tensor_filter_transform_get_invoke_tensors (GstBaseTransform *trans, FilterTransformData *trans_data)

  Internal function to get invoke tensors.

- static FilterTransformData * _gst_tensor_filter_transform_get_output_data (GstBaseTransform *trans)

  Internal function to get output tensors.

- static GstFlowReturn _gst_tensor_filter_transform_prepare_output_tensors (GstBaseTransform *trans, FilterTransformData *trans_data)

  Internal function to get output tensors.

- static GstFlowReturn _gst_tensor_filter_transform_check_invoke_result (GstBaseTransform *trans, FilterTransformData *in_trans_data, FilterTransformData *out_trans_data, gint invoke_res)

  Internal function to check the invoke result.

- static void _gst_tensor_filter_transform_update_outbuf (GstBaseTransform *trans, FilterTransformData *in↩ _trans_data, FilterTransformData *out_trans_data, GstBuffer *outbuf)

  Internal function to make output buffer.

- static gboolean gst_tensor_filter_watchdog_trigger (gpointer ptr)

  Called when there is no input within suspend time specified by the user.

- static gboolean gst_tensor_filter_configure_tensor (GstTensorFilter *self, const GstCaps *incaps)

  Configure input and output tensor info from incaps.

Variables

- static GstStaticPadTemplate sink_factory

  The capabilities of the inputs.

- static GstStaticPadTemplate src_factory

  The capabilities of the outputs.

9.104.1 Detailed Description

GStreamer plugin to use general neural network frameworks as filters.

GStreamer Tensor_Filter Copyright (C) 2005 Thomas Vander Stichele thomas@apestaart.org Copyright (C) 2005 Ronald S. Bultje rbultje@ronald.bitfreak.net Copyright (C) 2018 MyungJoo Ham myungjoo.ham@samsung.com

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Library General Public License as published by the Free Software Foundation; version 2.1 of the License.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Library General Public License for more details.

Date

24 May 2018

See also

http://github.com/nnstreamer/nnstreamer

Author

MyungJoo Ham myungjoo.ham@samsung.com

Bug No known bugs except for NYI items

Todo set priority among properties

logic for dynamic properties(like model change)

This is the main plugin for per-NN-framework plugins. Specific implementations for each NN framework must be written in each framework specific files; e.g., tensor_filter_tensorflow_lite.c

9.104.2 Macro Definition Documentation

9.104.2.1 CAPS_STRING

```
#define CAPS_STRING GST_TENSOR_CAP_DEFAULT ";" GST_TENSORS_CAP_MAKE ("{ static, flexible }")
```

Default caps string for both sink and source pad.

Definition at line 87 of file tensor_filter.c.

9.104.2.2 EVENT_NAME_UPDATE_MODEL

```
#define EVENT_NAME_UPDATE_MODEL "evt_update_model"
```

SECTION:element-tensor_filter

A plugin that invokes neural network models and their framework or an independent shared object implementing tensor_filter_custom.h. The input and output are always in the format of other/tensor or other/tensors.

<refsect2> <title>Example launch line</title> |[gst-launch -v -m fakesrc ! tensor_filter framework=tensorflow-lite, model=./inception_v3.pb, input=3:224:224, output=1000 ! fakesink silent=TRUE]| </refsect2>

If input is other/tensor C array input[1][224][224][3] and output is other/tensor C array output[1][1][1][1000]

The current QoS policy: In a nnstreamer pipeline, the QoS is currently satisfied by adjusting a input or output framerate, initiated by 'tensor_rate' element. When 'tensor_filter' receives a throttling QoS event from the 'tensor_↩ rate' element, it compares the average processing latency and throttling delay, and takes the maximum value as the threshold to drop incoming frames by checking a buffer timestamp. In this way, 'tensor filter' can avoid unnecessary calculation and adjust a framerate, effectively reducing resource utilizations. Even in the case of receiving QoS events from multiple downstream pipelines (e.g., tee), 'tensor_filter' takes the minimum value as the throttling delay for downstream pipeline with more tight QoS requirement. Lastly, 'tensor_filter' also sends QoS events to upstream elements (e.g., tensor_converter, tensor_src) to possibly reduce incoming framerates, which is a better solution than dropping framerates.

Todo rename & move this to better location

Definition at line 76 of file tensor_filter.c.

9.104.2.3 GST_CAT_DEFAULT

```
#define GST_CAT_DEFAULT gst_tensor_filter_debug
```

Definition at line 79 of file tensor_filter.c.

9.104.2.4 gst_tensor_filter_parent_class

```
#define gst_tensor_filter_parent_class parent_class
```

Definition at line 105 of file tensor_filter.c.

9.104.2.5 LATENCY_REPORT_HEADROOM

```
#define LATENCY_REPORT_HEADROOM 0.05
```

Headroom (extra duration) added to actual latency estimate reported to LATENCY query, to limit number of updates when tracking the maximum value - arbitrarily set to 5%.

Definition at line 113 of file tensor_filter.c.

9.104.2.6 LATENCY_REPORT_THRESHOLD

```
#define LATENCY_REPORT_THRESHOLD 0.25
```

Threshold deciding when tracking latency estimate that current value is sufficiently lower than reported value so that a notification update is necessary - arbitrarily set to 25%.

Definition at line 120 of file tensor_filter.c.

9.104.2.7 TF_MODELNAME

```
#define TF_MODELNAME(
              prop ) ((prop)->model_files ?  ((prop)->model_files[0]) :  "[No Model File]")
```

Definition at line 81 of file tensor_filter.c.

9.104.2.8 THRESHOLD_CACHE_OLD

```
#define THRESHOLD_CACHE_OLD (1000)
```

Definition at line 406 of file tensor_filter.c.

9.104.2.9 THRESHOLD_DROP_OLD

```
#define THRESHOLD_DROP_OLD (2000)
```

Definition at line 405 of file tensor_filter.c.

9.104.3 Typedef Documentation

9.104.3.1 FilterTransformData

```
typedef struct _FilterTransformData FilterTransformData
```

Internal data structure for tensor_filter transform data.

9.104.4 Function Documentation

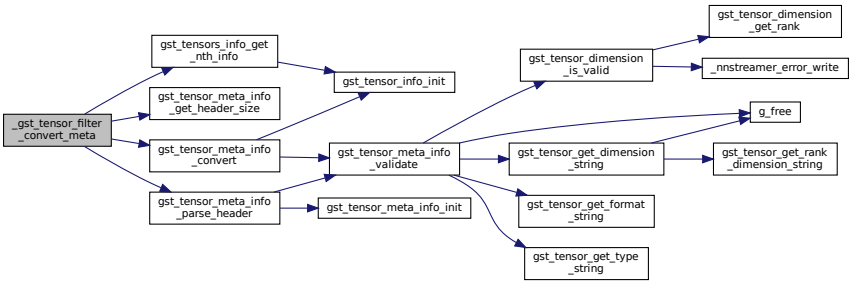
9.104.4.1 _gst_tensor_filter_convert_meta()

```
static gsize _gst_tensor_filter_convert_meta (
            FilterTransformData * trans_data,
            GstTensorsInfo * info,
            guint idx )  [static]
```

Internal function to convert tensor meta and get header size of flexible tensor.

Definition at line 684 of file tensor_filter.c.

Here is the call graph for this function:



Here is the caller graph for this function:

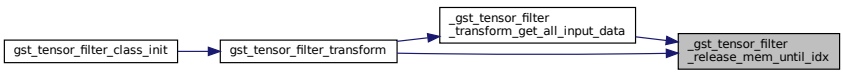
9.104.4.2 _gst_tensor_filter_release_mem_until_idx()

```
static void _gst_tensor_filter_release_mem_until_idx (
            FilterTransformData * trans_data,
            guint end_index )  [static]
```

Internal function to release mem and unmap info with index.

Definition at line 667 of file tensor_filter.c.

Here is the caller graph for this function:



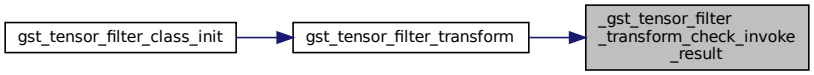
9.104.4.3 _gst_tensor_filter_transform_check_invoke_result()

```
static GstFlowReturn _gst_tensor_filter_transform_check_invoke_result (
            GstBaseTransform * trans,
            FilterTransformData * in_trans_data,
            FilterTransformData * out_trans_data,
            gint invoke_res )  [static]
```

Internal function to check the invoke result.

Definition at line 936 of file tensor_filter.c.

Here is the caller graph for this function:



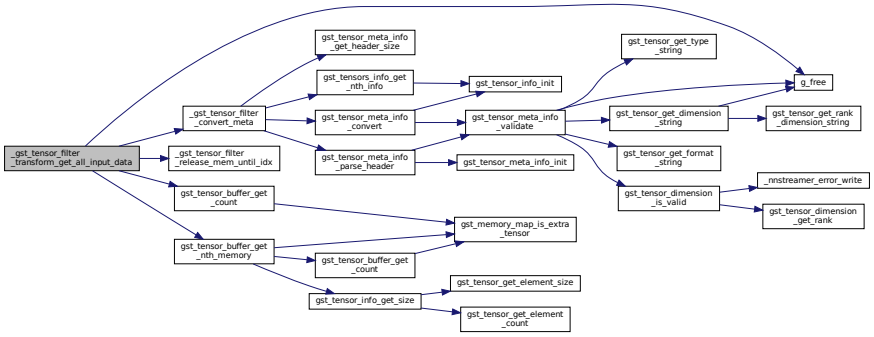
9.104.4.4 _gst_tensor_filter_transform_get_all_input_data()

```
static FilterTransformData* _gst_tensor_filter_transform_get_all_input_data (
            GstBaseTransform * trans,
            GstBuffer * buf )  [static]
```

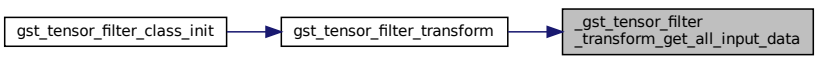
Internal function to get input tensors.

Definition at line 707 of file tensor_filter.c.

Here is the call graph for this function:



Here is the caller graph for this function:



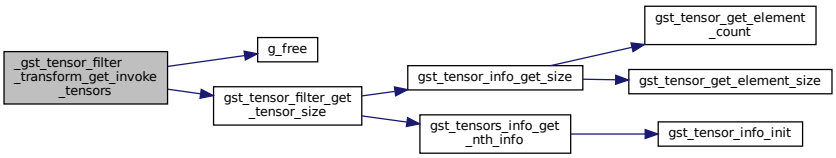
9.104.4.5 _gst_tensor_filter_transform_get_invoke_tensors()

```
static GstTensorMemory* _gst_tensor_filter_transform_get_invoke_tensors (
            GstBaseTransform * trans,
            FilterTransformData * trans_data )  [static]
```

Internal function to get invoke tensors.

Definition at line 754 of file tensor_filter.c.

Here is the call graph for this function:



Here is the caller graph for this function:

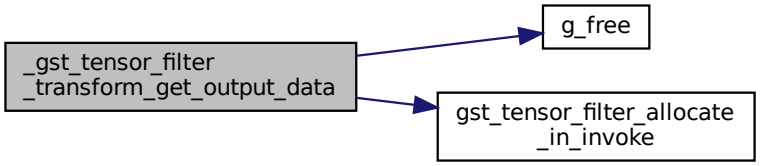
9.104.4.6 _gst_tensor_filter_transform_get_output_data()

```
static FilterTransformData* _gst_tensor_filter_transform_get_output_data (
            GstBaseTransform * trans )  [static]
```

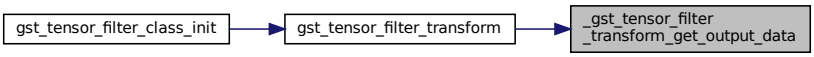
Internal function to get output tensors.

Definition at line 836 of file tensor_filter.c.

Here is the call graph for this function:



Here is the caller graph for this function:



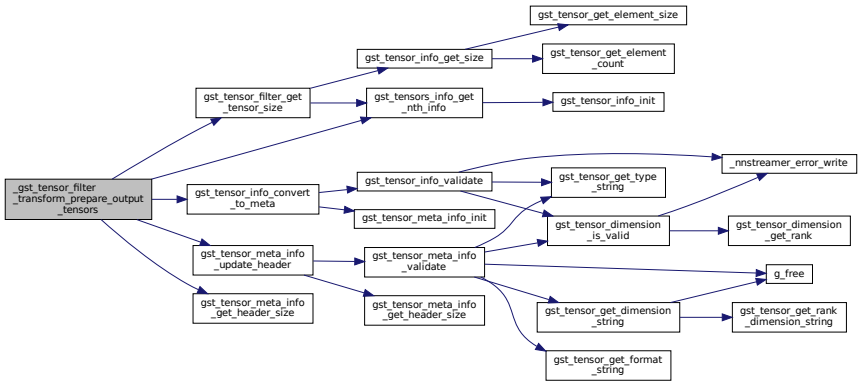
9.104.4.7 _gst_tensor_filter_transform_prepare_output_tensors()

```
static GstFlowReturn _gst_tensor_filter_transform_prepare_output_tensors (
            GstBaseTransform * trans,
            FilterTransformData * trans_data )  [static]
```

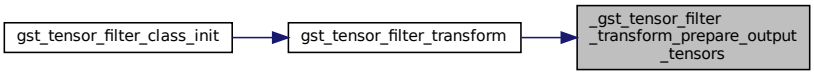
Internal function to get output tensors.

Definition at line 870 of file tensor_filter.c.

Here is the call graph for this function:



Here is the caller graph for this function:



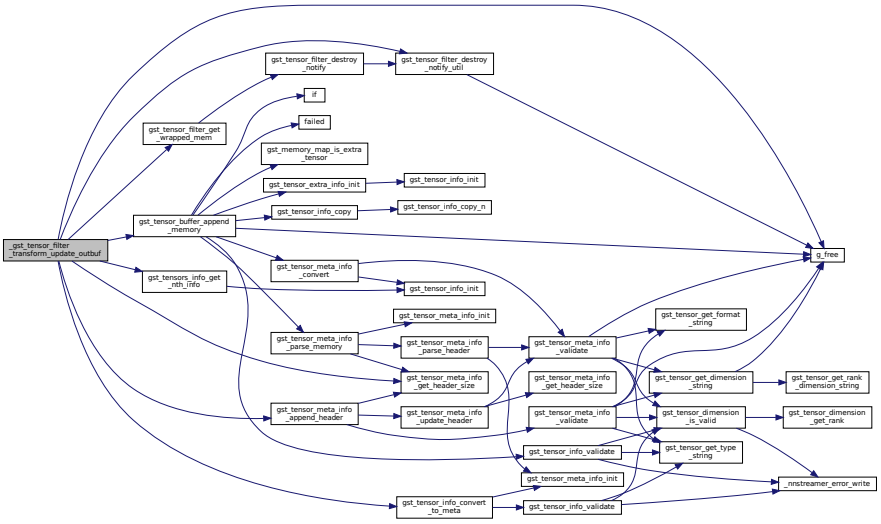
9.104.4.8 _gst_tensor_filter_transform_update_outbuf()

```
static void _gst_tensor_filter_transform_update_outbuf (
            GstBaseTransform * trans,
            FilterTransformData * in_trans_data,
            FilterTransformData * out_trans_data,
            GstBuffer * outbuf )  [static]
```

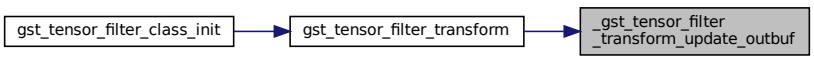
Internal function to make output buffer.

Definition at line 976 of file tensor_filter.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.104.4.9 _gst_tensor_filter_transform_validate()

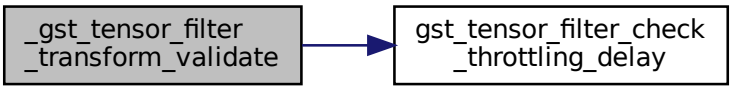
```
static GstFlowReturn _gst_tensor_filter_transform_validate (
            GstBaseTransform * trans,
            GstBuffer * inbuf,
            GstBuffer * outbuf )  [static]
```

Check input parameters for gst_tensor_filter_transform ();.

This is fatal; if framework is not configured until this stage, it means that an extension is missing or not configured. We need readable messages for non-developers

Definition at line 593 of file tensor_filter.c.

Here is the call graph for this function:



Here is the caller graph for this function:

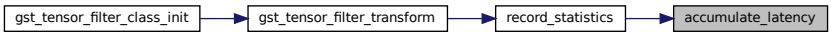
9.104.4.10 accumulate_latency()

```
static void accumulate_latency (
            void * data,
            void * user_data )  [static]
```

Helper function to accumulate latencies.

Definition at line 397 of file tensor_filter.c.

Here is the caller graph for this function:



9.104.4.11 G_DEFINE_TYPE()

```
G_DEFINE_TYPE (
             GstTensorFilter ,
             gst_tensor_filter ,
             GST_TYPE_BASE_TRANSFORM  )
```

9.104.4.12 GST_DEBUG_CATEGORY_STATIC()

```
GST_DEBUG_CATEGORY_STATIC (
             gst_tensor_filter_debug  )
```

9.104.4.13 gst_tensor_filter_check_throttling_delay()

```
static gboolean gst_tensor_filter_check_throttling_delay (
             GstBaseTransform * trans,
             GstBuffer * inbuf )  [static]
```

Check throttling delay and send qos overflow event to upstream elements.

Send qos overflow event to upstream elements. Upstream elements (e.g., tensor_src, tensor_converter) may handle this.

Definition at line 536 of file tensor_filter.c.

Here is the caller graph for this function:

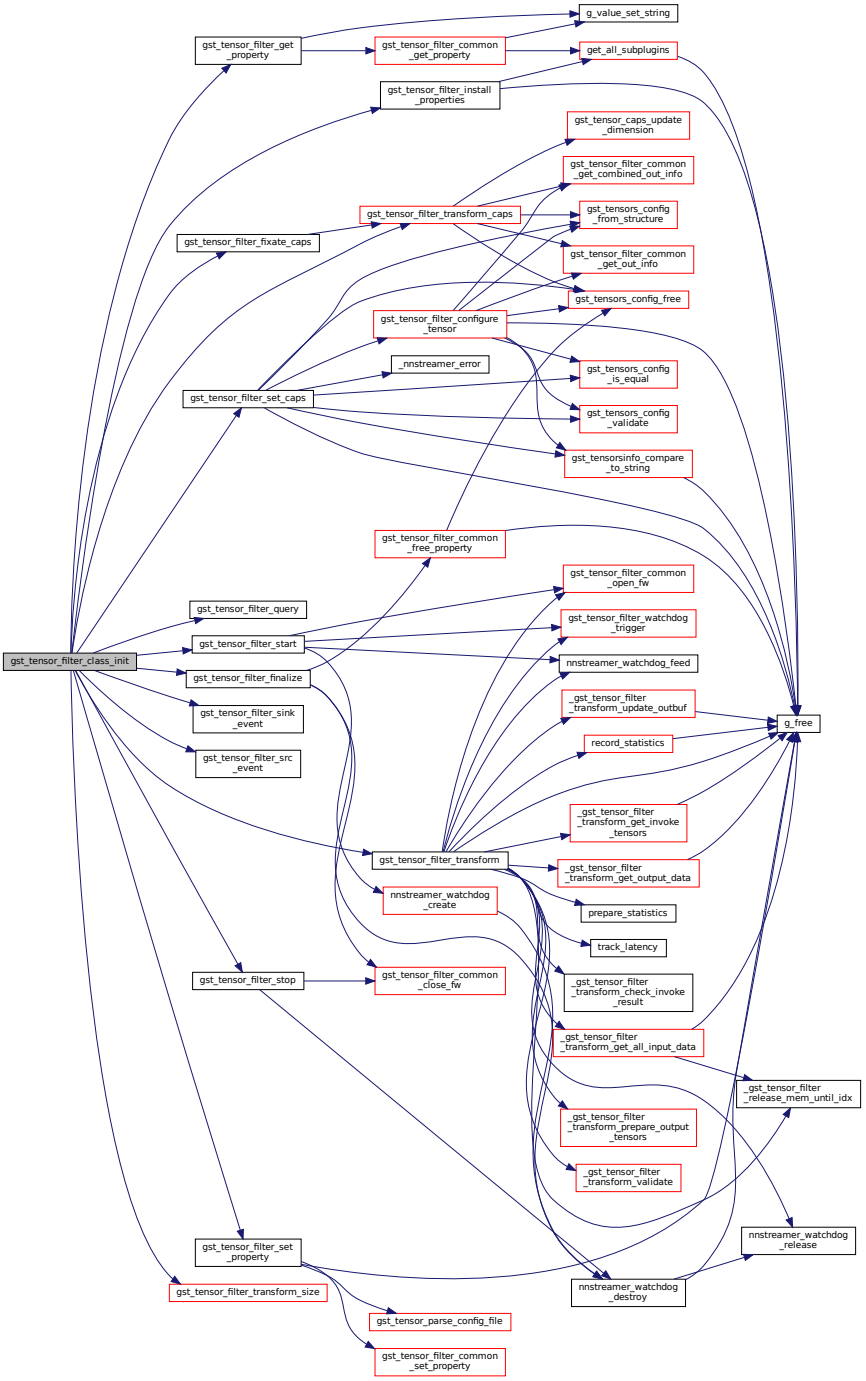
9.104.4.14 gst_tensor_filter_class_init()

```
static void gst_tensor_filter_class_init (
            GstTensorFilterClass * klass )  [static]
```

initialize the tensor_filter's class

Definition at line 169 of file tensor_filter.c.

Here is the call graph for this function:



9.104.4.15 gst_tensor_filter_configure_tensor()

```
static gboolean gst_tensor_filter_configure_tensor (
            GstTensorFilter * self,
            const GstCaps * incaps )  [static]
```

Configure input and output tensor info from incaps.

Parameters

self	"this" pointer
incaps	received caps for sink pad

Returns

TRUE if fully configured

GstTensorFilter has to parse the tensor dimension and type from NN model.

1. Call functions getInputDimension and getOutputDimension to get the dimension and type.
2. If these functions are not defined, call setInputDimension with parsed info from caps.
3. If set-prop configured dimension, verify the dimension with fw callbacks.

Check configuration from caps. If true, fully configured tensor info from caps.

if set-property called and already has info, verify it!

If incoming tensor is flexible, we cannot validate tensor info here. Need to compare buffer size in transform().

Todo We do not support this (flexible tensor for flexible input model). Cap-negotiation of the current tensor-filter requires either side of "model / set-property" or "incoming gstcaps" to be static/explicit. Ideally, this should support flexible tensor for flexible input model, leaving the negotiation to other elements, but we didn't implement it yet.

call setInputDimension if output tensor is not configured

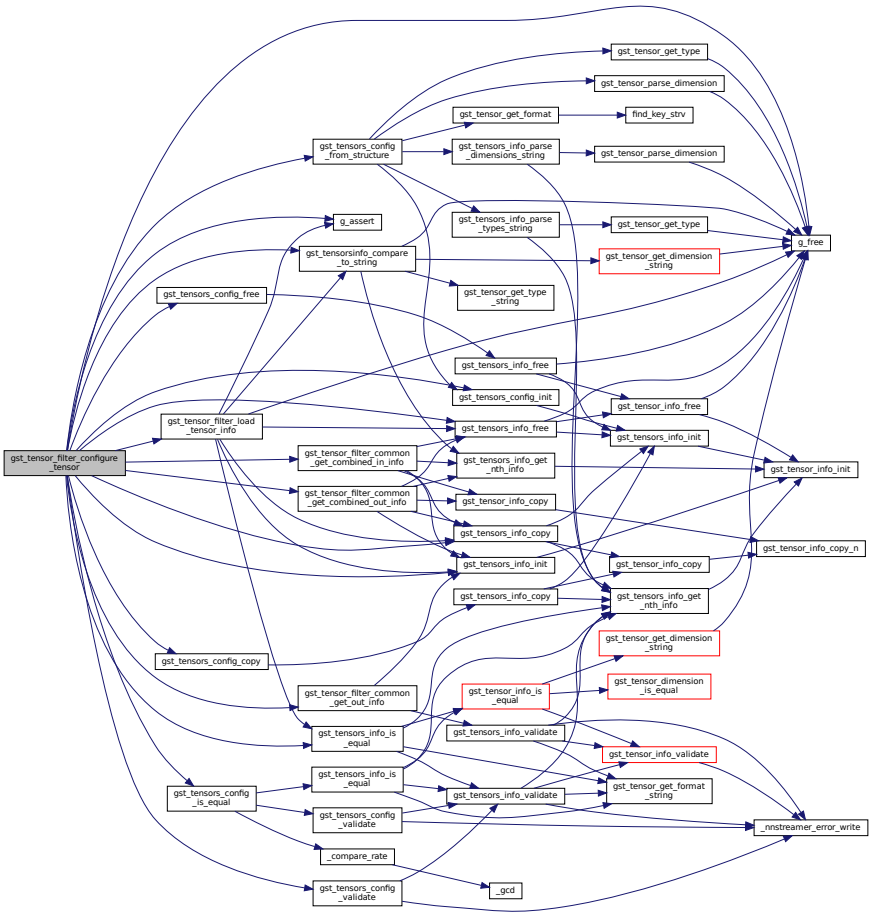
if set-property called and already has info, verify it!

Todo framerate of output tensors How can we update the framerate? GstTensorFilter cannot assure the framerate. Simply set the framerate of out-tensor from incaps.

already configured, compare to old.

Definition at line 1207 of file tensor_filter.c.

Here is the call graph for this function:



Here is the caller graph for this function:

9.104.4.16 gst_tensor_filter_destroy_notify()

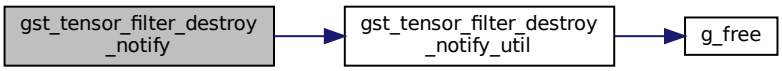
```
static void gst_tensor_filter_destroy_notify (
            void * data )  [static]
```

Free the data allocated for tensor transform.

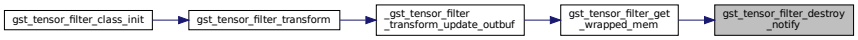
default function for tensor filter framework if not provided by the framework. The data is in GPtrArray - first element is private data of framework and second element is the data to be freed.

Definition at line 357 of file tensor_filter.c.

Here is the call graph for this function:



Here is the caller graph for this function:



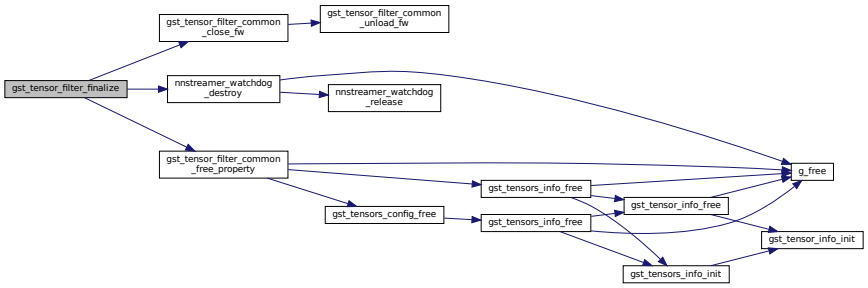
9.104.4.17 gst_tensor_filter_finalize()

```
static void gst_tensor_filter_finalize (
            GObject * object )  [static]
```

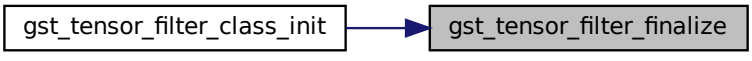
Function to finalize instance.

Definition at line 249 of file tensor_filter.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.104.4.18 gst_tensor_filter_fixate_caps()

```
static GstCaps * gst_tensor_filter_fixate_caps (
             GstBaseTransform * trans,
             GstPadDirection direction,
             GstCaps * caps,
             GstCaps * othercaps )  [static]
```

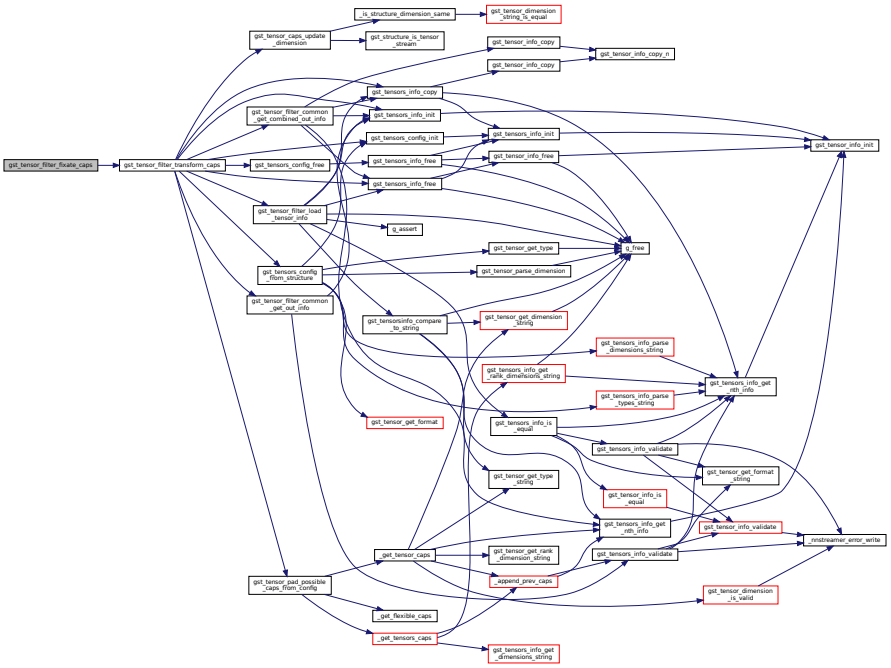
fixate caps. required vmethod of GstBaseTransform.

Removes no-used-variable warning for priv in when DBG is set

To get the out-caps, GstTensorFilter has to parse tensor info from NN model.

Definition at line 1528 of file tensor_filter.c.

Here is the call graph for this function:



Here is the caller graph for this function:

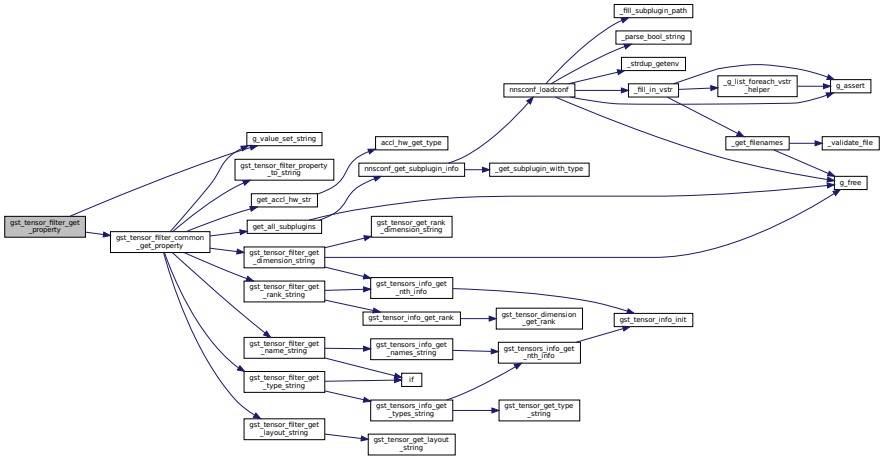
9.104.4.19 gst_tensor_filter_get_property()

```
static void gst_tensor_filter_get_property (
            GObject * object,
            guint prop_id,
            GValue * value,
            GParamSpec * pspec )  [static]
```

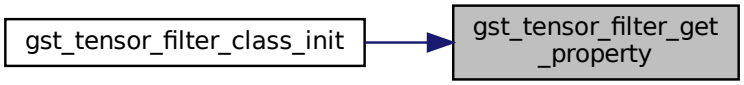
Getter for tensor_filter properties.

Definition at line 330 of file tensor_filter.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.104.4.20 gst_tensor_filter_get_tensor_size()

```
static gsize gst_tensor_filter_get_tensor_size (
            GstTensorFilter * self,
            guint index,
            gboolean is_input )  [static]
```

Calculate tensor buffer size.

Parameters

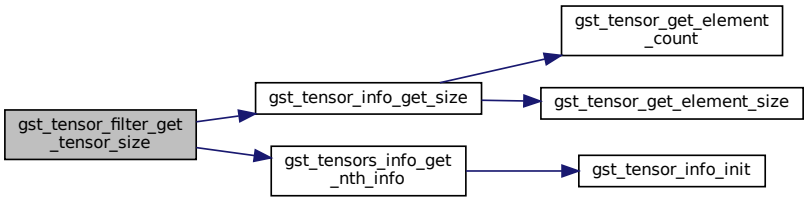
self	"this" pointer
index	index of tensors

Returns

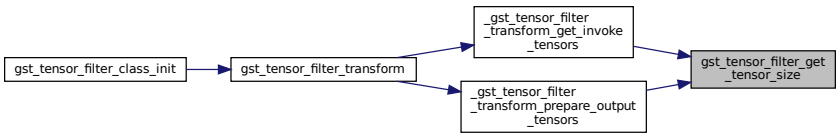
tensor buffer size

Definition at line 277 of file tensor_filter.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.104.4.21 gst_tensor_filter_get_wrapped_mem()

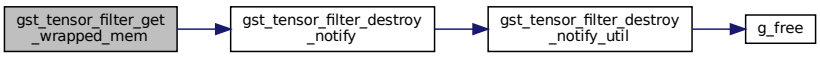
```
static GstMemory* gst_tensor_filter_get_wrapped_mem (
            GstTensorFilter * self,
            gpointer data,
            gsize size )  [static]
```

Allocate new memory block from given data.

tensor-filter should send event to sub-plugin when memory is freed.

Definition at line 372 of file tensor_filter.c.

Here is the call graph for this function:



Here is the caller graph for this function:

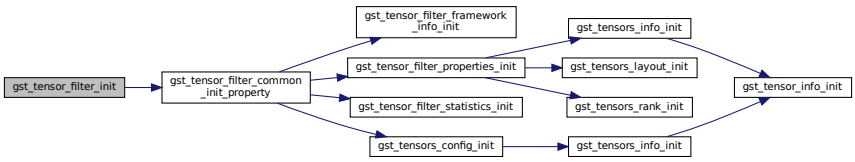
9.104.4.22 gst_tensor_filter_init()

```
static void gst_tensor_filter_init (
            GstTensorFilter * self )  [static]
```

initialize the new element instantiate pads and add them to element set pad callback functions initialize instance structure

Definition at line 234 of file tensor_filter.c.

Here is the call graph for this function:



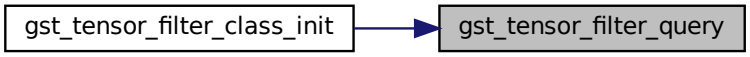
9.104.4.23 gst_tensor_filter_query()

```
static gboolean gst_tensor_filter_query (
            GstBaseTransform * trans,
            GstPadDirection direction,
            GstQuery * query )  [static]
```

query handling, optional vmethod of GstBaseTransform.

Definition at line 1623 of file tensor_filter.c.

Here is the caller graph for this function:



9.104.4.24 gst_tensor_filter_set_caps()

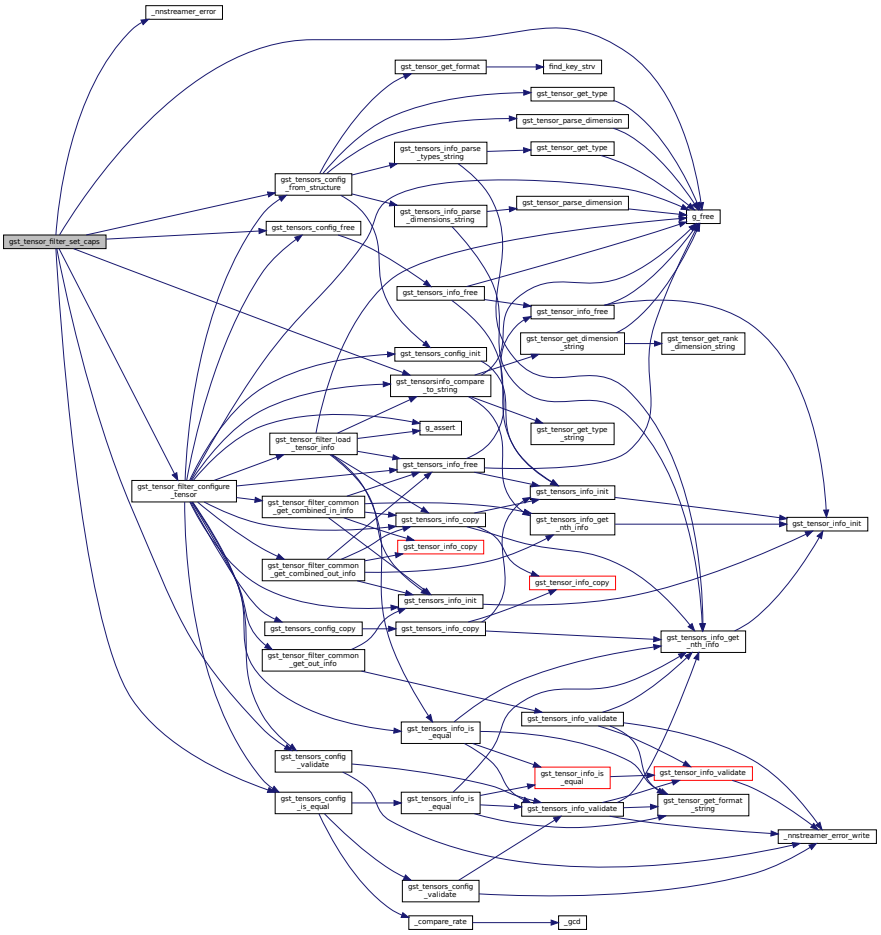
```
static gboolean gst_tensor_filter_set_caps (
            GstBaseTransform * trans,
            GstCaps * incaps,
            GstCaps * outcaps )  [static]
```

set caps. required vmethod of GstBaseTransform.

compare output tensor

Definition at line 1563 of file tensor_filter.c.

Here is the call graph for this function:



Here is the caller graph for this function:

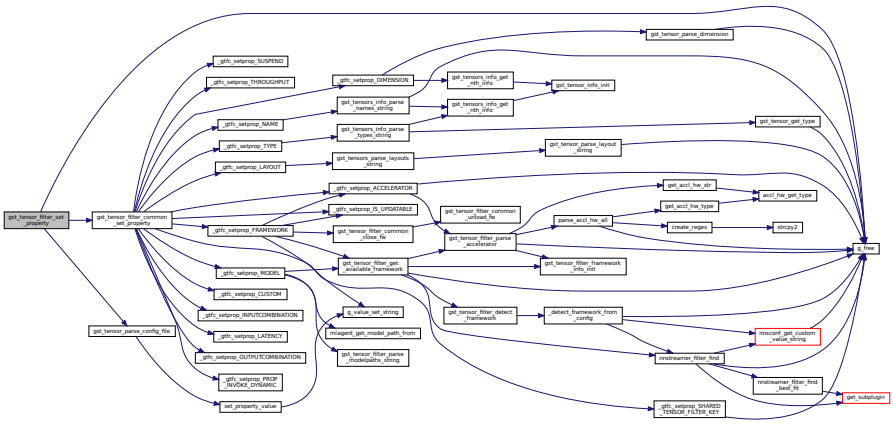
9.104.4.25 gst_tensor_filter_set_property()

```
static void gst_tensor_filter_set_property (
            GObject * object,
            guint prop_id,
            const GValue * value,
            GParamSpec * pspec )  [static]
```

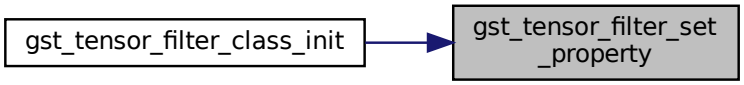
Setter for tensor_filter properties.

Definition at line 304 of file tensor_filter.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.104.4.26 gst_tensor_filter_sink_event()

```
static gboolean gst_tensor_filter_sink_event (
            GstBaseTransform * trans,
            GstEvent * event )  [static]
```

Event handler for sink pad of tensor filter.

Parameters

trans	"this" pointer
event	a passed event object

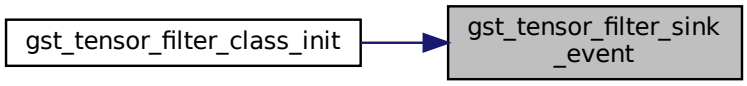
Returns

TRUE if there is no error.

other events are handled in the default event handler

Definition at line 1723 of file tensor_filter.c.

Here is the caller graph for this function:



9.104.4.27 gst_tensor_filter_src_event()

```
static gboolean gst_tensor_filter_src_event (
            GstBaseTransform * trans,
            GstEvent * event )  [static]
```

Event handler for src pad of tensor filter.

Parameters

trans	"this" pointer
event	a passed event object

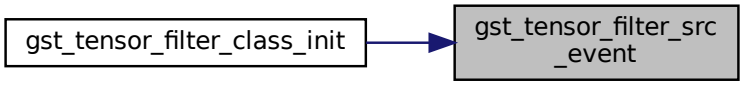
Returns

TRUE if there is no error.

other events are handled in the default event handler

Definition at line 1764 of file tensor_filter.c.

Here is the caller graph for this function:



9.104.4.28 gst_tensor_filter_start()

```
static gboolean gst_tensor_filter_start (
            GstBaseTransform * trans )  [static]
```

Called when the element starts processing. optional vmethod of BaseTransform.

Parameters

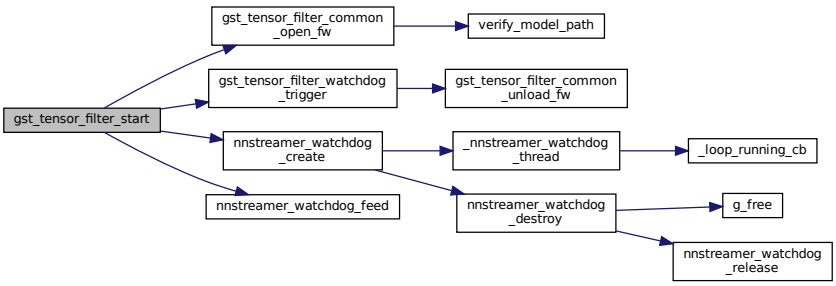
trans	"this" pointer

Returns

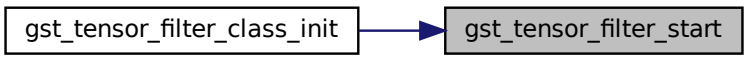
TRUE if there is no error.

Definition at line 1802 of file tensor_filter.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.104.4.29 gst_tensor_filter_stop()

```
static gboolean gst_tensor_filter_stop (
            GstBaseTransform * trans )  [static]
```

Called when the element stops processing. optional vmethod of BaseTransform.

Parameters

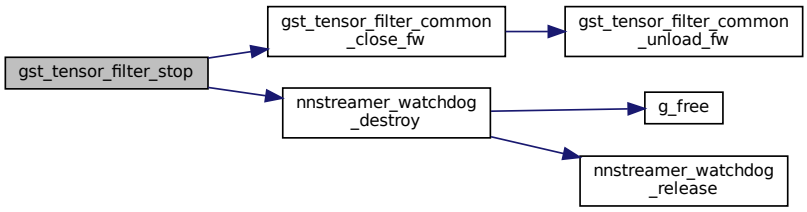
trans	"this" pointer

Returns

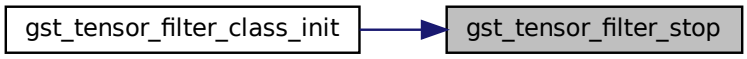
TRUE if there is no error.

Definition at line 1836 of file tensor_filter.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.104.4.30 gst_tensor_filter_transform()

```
static GstFlowReturn gst_tensor_filter_transform (
            GstBaseTransform * trans,
            GstBuffer * inbuf,
            GstBuffer * outbuf )  [static]
```

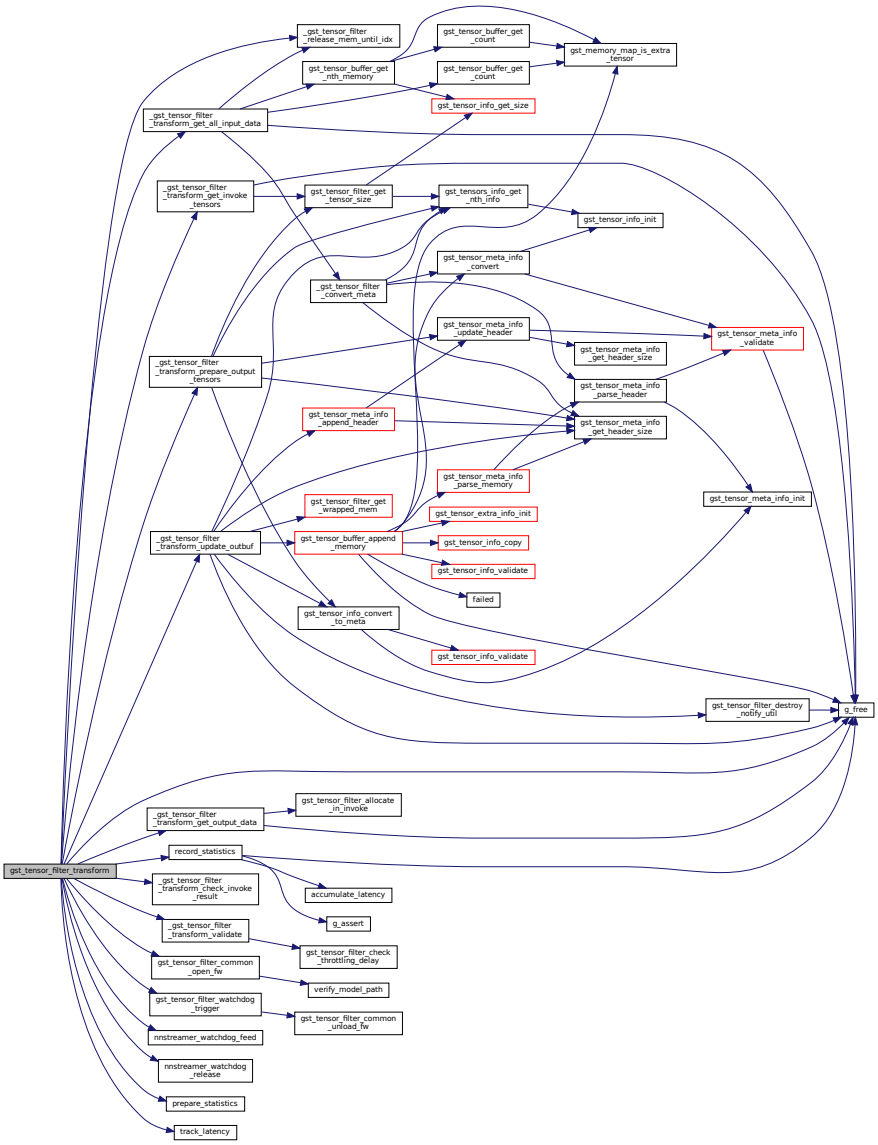
non-ip transform. required vmethod of GstBaseTransform.

Reset suspend timeout

Set suspend timeout

Definition at line 1097 of file tensor_filter.c.

Here is the call graph for this function:



Here is the caller graph for this function:

9.104.4.31 gst_tensor_filter_transform_caps()

```
static GstCaps * gst_tensor_filter_transform_caps (
            GstBaseTransform * trans,
            GstPadDirection direction,
            GstCaps * caps,
            GstCaps * filter )  [static]
```

configure tensor-srcpad cap from "proposed" cap.

@trans ("this" pointer) @direction (why do we need this?) @caps sinkpad cap (if direction GST_PAD_SINK) @filter this element's cap (don't know specifically.)

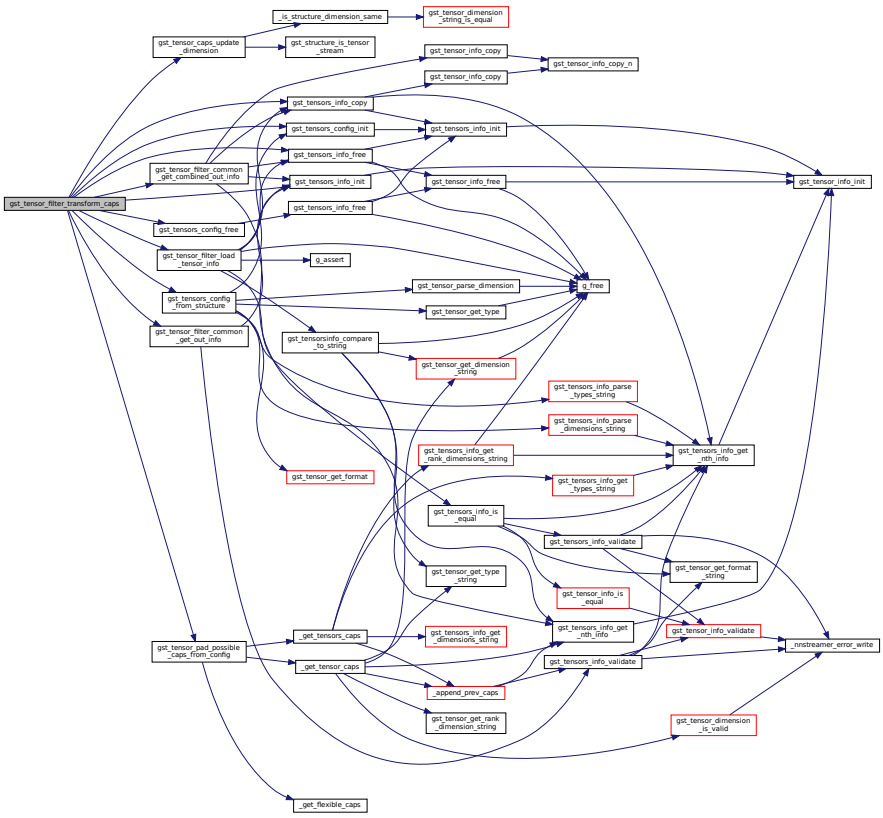
Be careful not to fix/set caps at this stage. Negotiation not completed yet. GstTensorFilter has to parse the tensor dimension and type from NN model. In this stage, in-caps may not be fixed yet. To get the tensor info and generate pad-caps, call getInputDimension and getOutputDimension. If these functions are not defined, we have to call setInputDimension, and then it will fully configure the tensor info.

Todo how to set the framerate of output tensors

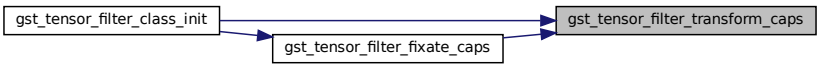
Todo We do not have a testcase hitting here. Thus, we do not ensure the validity here. However, according to gstreamer doxygen entry, if filter is given, that's not to be ignored. For now, we assume that if caps-size is 0, filter is "ANY".

Definition at line 1405 of file tensor_filter.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.104.4.32 gst_tensor_filter_transform_size()

```
static gboolean gst_tensor_filter_transform_size (
            GstBaseTransform * trans,
            GstPadDirection direction,
            GstCaps * caps,
            gsize size,
            GstCaps * othercaps,
            gsize * othersize )  [static]
```

Tell the framework the required size of buffer based on the info of the other side pad. optional vmethod of Base↩ Transform.

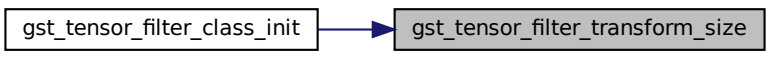
We cannot directly get the value from size value, we need to review the pad-caps. This is called when non-ip mode is used. Internal Logic Error. Cannot proceed without configured pipeline

Consider multi-tensors. Set each memory block in transform()

Definition at line 1694 of file tensor_filter.c.

Here is the call graph for this function:

Here is the caller graph for this function:



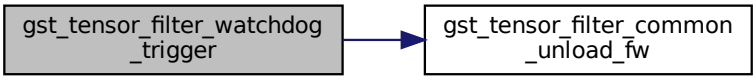
9.104.4.33 gst_tensor_filter_watchdog_trigger()

```
static gboolean gst_tensor_filter_watchdog_trigger (
            gpointer ptr )  [static]
```

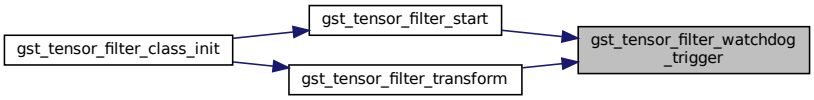
Called when there is no input within suspend time specified by the user.

Definition at line 1083 of file tensor_filter.c.

Here is the call graph for this function:



Here is the caller graph for this function:



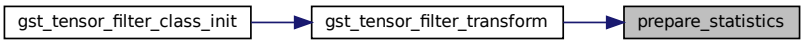
9.104.4.34 prepare_statistics()

```
static void prepare_statistics (
            GstTensorFilterPrivate * priv )  [static]
```

Prepare statistics for performance profiling (e.g, latency, throughput)

Definition at line 388 of file tensor_filter.c.

Here is the caller graph for this function:



9.104.4.35 record_statistics()

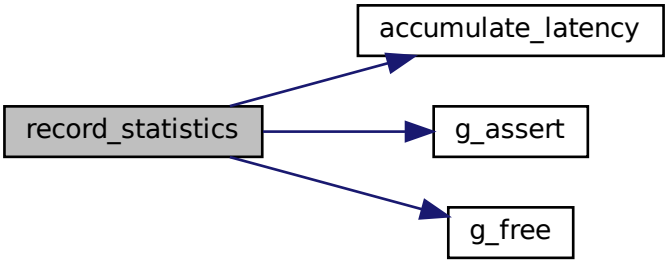
```
static void record_statistics (
            GstTensorFilterPrivate * priv )  [static]
```

Record statistics for performance profiling (e.g, latency, throughput)

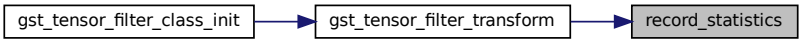
statistics values are monotonously increasing. to avoid potential overflow, let's cache old values and subtract them from the statistics if some threshold is exceeded.

Definition at line 412 of file tensor_filter.c.

Here is the call graph for this function:



Here is the caller graph for this function:



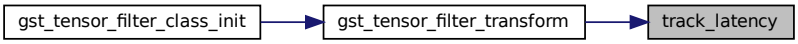
9.104.4.36 track_latency()

```
static void track_latency (
            GstTensorFilter * self )  [static]
```

Track estimated latency and notify pipeline when it changes. Latency estimates may be a bit jittery. On the principle we want to inform pipeline with the latency from longest inference. However, first inference may take much longer, or model filter configuration may change. Therefore any change of more than 10% (arbitrary value) to a lower latency is also reported to pipeline. Notification is done sending LATENCY message to bus. Upon receipt, application will initiate a pipeline latency probe via LATENCY query.

Definition at line 503 of file tensor_filter.c.

Here is the caller graph for this function:



9.104.5 Variable Documentation

9.104.5.1 sink_factory

```
GstStaticPadTemplate sink_factory  [static]
```

Initial value:

```
= GST_STATIC_PAD_TEMPLATE ("sink",
    GST_PAD_SINK,
    GST_PAD_ALWAYS,
    GST_STATIC_CAPS (CAPS_STRING))
```

The capabilities of the inputs.

Definition at line 92 of file tensor_filter.c.

9.104.5.2 src_factory

```
GstStaticPadTemplate src_factory  [static]
```

Initial value:

```
= GST_STATIC_PAD_TEMPLATE ("src",
    GST_PAD_SRC,
    GST_PAD_ALWAYS,
    GST_STATIC_CAPS (CAPS_STRING))
```

The capabilities of the outputs.

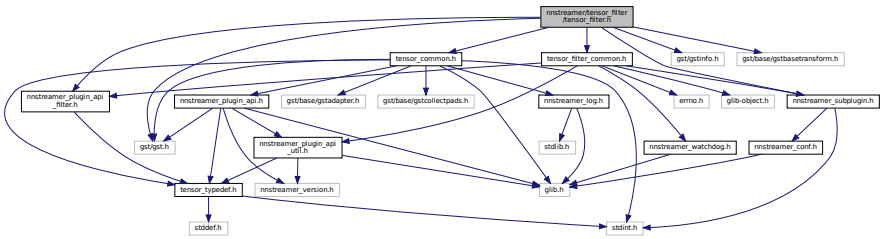
Definition at line 100 of file tensor_filter.c.

9.105 nnstreamer/tensor_filter/tensor_filter.h File Reference

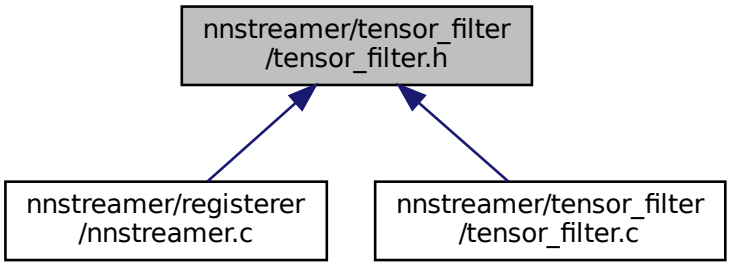
GStreamer plugin to use general neural network frameworks as filters.

```
#include <gst/gst.h>
#include <gst/gstinfo.h>
#include <gst/base/gstbasetransform.h>
#include "tensor_common.h"
#include "nnstreamer_subplugin.h"
#include "nnstreamer_plugin_api_filter.h"
#include "tensor_filter_common.h"
```

Include dependency graph for tensor_filter.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct _GstTensorFilter

  Internal data structure for tensor_filter instances.
- struct _GstTensorFilterClass

  GstTensorFilterClass inherits GstBaseTransformClass.

Macros

- #define GST_TYPE_TENSOR_FILTER (gst_tensor_filter_get_type())
- #define GST_TENSOR_FILTER(obj) (G_TYPE_CHECK_INSTANCE_CAST((obj),GST_TYPE_TENSOR_FILTER,GstTensorF
- #define GST_TENSOR_FILTER_CLASS(klass) (G_TYPE_CHECK_CLASS_CAST((klass),GST_TYPE_TENSOR_FILTER,Gs
- #define GST_IS_TENSOR_FILTER(obj) (G_TYPE_CHECK_INSTANCE_TYPE((obj),GST_TYPE_TENSOR_FILTER))
- #define GST_IS_TENSOR_FILTER_CLASS(klass) (G_TYPE_CHECK_CLASS_TYPE((klass),GST_TYPE_TENSOR_FILTER
- #define GST_TENSOR_FILTER_CAST(obj) ((GstTensorFilter *)(obj))

Typedefs

- typedef struct _GstTensorFilter GstTensorFilter
- typedef struct _GstTensorFilterClass GstTensorFilterClass

Functions

- GType gst_tensor_filter_get_type (void)

  Get Type function required for gst elements.

9.105.1 Detailed Description

GStreamer plugin to use general neural network frameworks as filters.

GStreamer Tensor_Filter Copyright (C) 2005 Thomas Vander Stichele thomas@apestaart.org Copyright (C) 2005 Ronald S. Bultje rbultje@ronald.bitfreak.net Copyright (C) 2018 MyungJoo Ham myungjoo.ham@samsung.com

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Library General Public License as published by the Free Software Foundation; version 2.1 of the License.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Library General Public License for more details.

Date

24 May 2018

See also

http://github.com/nnstreamer/nnstreamer

Author

MyungJoo Ham myungjoo.ham@samsung.com

Bug No known bugs except for NYI items

Todo TBD: Should we disable "in-place" mode? (what if output size > input size?)

9.105.2 Macro Definition Documentation

9.105.2.1 GST_IS_TENSOR_FILTER

```
#define GST_IS_TENSOR_FILTER(
              obj ) (G_TYPE_CHECK_INSTANCE_TYPE((obj),GST_TYPE_TENSOR_FILTER))
```

Definition at line 49 of file tensor_filter.h.

9.105.2.2 GST_IS_TENSOR_FILTER_CLASS

```
#define GST_IS_TENSOR_FILTER_CLASS(
              klass ) (G_TYPE_CHECK_CLASS_TYPE((klass),GST_TYPE_TENSOR_FILTER))
```

Definition at line 51 of file tensor_filter.h.

9.105.2.3 GST_TENSOR_FILTER

```
#define GST_TENSOR_FILTER(
              obj ) (G_TYPE_CHECK_INSTANCE_CAST((obj),GST_TYPE_TENSOR_FILTER,GstTensorFilter))
```

Definition at line 45 of file tensor_filter.h.

9.105.2.4 GST_TENSOR_FILTER_CAST

```
#define GST_TENSOR_FILTER_CAST(
              obj ) ((GstTensorFilter *)(obj))
```

Definition at line 53 of file tensor_filter.h.

9.105.2.5 GST_TENSOR_FILTER_CLASS

```
#define GST_TENSOR_FILTER_CLASS(
              klass ) (G_TYPE_CHECK_CLASS_CAST((klass),GST_TYPE_TENSOR_FILTER,GstTensorFilterClass))
```

Definition at line 47 of file tensor_filter.h.

9.105.2.6 GST_TYPE_TENSOR_FILTER

```
#define GST_TYPE_TENSOR_FILTER (gst_tensor_filter_get_type())
```

Definition at line 43 of file tensor_filter.h.

9.105.3 Typedef Documentation

9.105.3.1 GstTensorFilter

```
typedef struct _GstTensorFilter GstTensorFilter
```

Definition at line 55 of file tensor_filter.h.

9.105.3.2 GstTensorFilterClass

```
typedef struct _GstTensorFilterClass GstTensorFilterClass
```

Definition at line 56 of file tensor_filter.h.

9.105.4 Function Documentation

9.105.4.1 gst_tensor_filter_get_type()

```
GType gst_tensor_filter_get_type (
            void  )
```

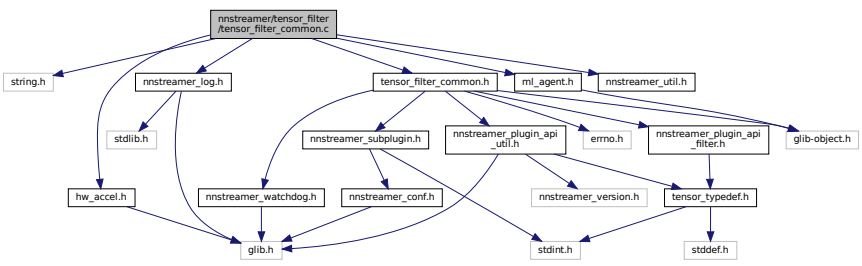
Get Type function required for gst elements.

9.106 nnstreamer/tensor_filter/tensor_filter_common.c File Reference

Common functions for various tensor_filters.

```
#include <string.h>
#include <hw_accel.h>
#include <ml_agent.h>
#include <nnstreamer_log.h>
#include <nnstreamer_util.h>
#include "tensor_filter_common.h"
```

Include dependency graph for tensor_filter_common.c:



Macros

- #define silent_debug_info(i, msg)
- #define REGEX_ACCL_ELEM_START "("
- #define REGEX_ACCL_ELEM_PREFIX "(?<!!)"
- #define REGEX_ACCL_ELEM_SUFFIX ""
- #define REGEX_ACCL_ELEM_DELIMITER "|"
- #define REGEX_ACCL_ELEM_END ")?"
- #define REGEX_ACCL_START "(^(true)[:]?([(]?("
- #define REGEX_ACCL_PREFIX ""
- #define REGEX_ACCL_SUFFIX ""
- #define REGEX_ACCL_DELIMITER "|"
- #define REGEX_ACCL_END ")*[)]?))"
- #define g_free_const(x) g_free((void*)(long)(x))

  Free memory.
- #define g_strfreev_const(x) g_strfreev((void*)(long)(x))

Functions

- static GType accl_hw_get_type (void)

  to get and register hardware accelerator backend enum
- static GList * parse_accl_hw_all (const gchar *accelerators, const gchar **supported_accelerators)

  parse user given string to extract list of accelerators based on given regex
- static gint _gtfc_setprop_IS_UPDATABLE (GstTensorFilterPrivate *priv, GstTensorFilterProperties *prop, const GValue *value)

  Handle "PROP_IS_UPDATABLE" for set-property.
- static gint _gtfc_setprop_ACCELERATOR (GstTensorFilterPrivate *priv, GstTensorFilterProperties *prop, const GValue *value)

  Handle "PROP_ACCELERATOR" for set-property.
- G_LOCK_DEFINE_STATIC (shared_model_table)

  mutex for shared model table.
- static void gst_tensors_layout_init (tensors_layout layout)

  Initialize the tensors layout.
- static void gst_tensors_rank_init (unsigned int ranks[])

  Initialize the tensors ranks.
- static tensor_layout gst_tensor_parse_layout_string (const gchar *layoutstr)

  Get tensor layout from string input.
- static guint gst_tensors_parse_layouts_string (tensors_layout layout, const gchar *layout_string)

  Parse the string of tensor layouts.
- static gchar * gst_tensor_filter_get_rank_string (const GstTensorFilterProperties *prop, gboolean isInput)

  Get the rank string of the tensors.
- static gchar * gst_tensor_filter_get_dimension_string (const GstTensorFilterProperties *prop, const gboolean isInput)

  Get the dimension string of tensors considering rank count.
- static gchar * gst_tensor_filter_get_type_string (const GstTensorFilterProperties *prop, const gboolean is↩_input)

  Get the type string of tensors.
- static gchar * gst_tensor_filter_get_name_string (const GstTensorFilterProperties *prop, const gboolean is↩_input)

  Get the name string of tensors.
- static const gchar * gst_tensor_get_layout_string (tensor_layout layout)

  Get layout string of tensor layout.

- static gchar * gst_tensor_filter_get_layout_string (const GstTensorFilterProperties *prop, const gboolean is↩ _input)

  Get the string of layout of tensors.
- static gchar * strcpy2 (gchar *dest, const gchar *src)

  copy the string from src to destination
- static gchar * create_regex (const gchar **enum_list, const gchar **regex_utils)

  create regex for the given string list and regex basic elements
- static gboolean verify_model_path (const GstTensorFilterPrivate *priv)

  Verify validity of path for given model file if verify_model_path is set.
- static void gst_tensor_filter_properties_init (GstTensorFilterProperties *prop)

  Initialize the GstTensorFilterProperties object.
- static void gst_tensor_filter_framework_info_init (GstTensorFilterFrameworkInfo *info)

  Initialize the GstTensorFilterFrameworkInfo object.
- static void gst_tensor_filter_statistics_init (GstTensorFilterStatistics *stat)

  Initialize the GstTensorFilterFrameworkInfo object.
- static gboolean nnstreamer_filter_validate (const GstTensorFilterFramework *tfsp)

  Validate filter sub-plugin's data.
- int nnstreamer_filter_probe (GstTensorFilterFramework *tfsp)

  Filter's sub-plugin should call this function to register itself.
- void nnstreamer_filter_exit (const char *name)

  Filter's sub-plugin may call this to unregister itself.
- void nnstreamer_filter_set_custom_property_desc (const char *name, const char *prop,...)

  set custom property description for tensor filter sub-plugin
- static const GstTensorFilterFramework * nnstreamer_filter_find_best_fit (const char *names)

  Find sub-plugin filter given the name list.
- const GstTensorFilterFramework * nnstreamer_filter_find (const char *name)

  Find filter sub-plugin with the name.
- static void gst_tensor_filter_parse_modelpaths_string (GstTensorFilterProperties *prop, const gchar *model_files)

  Parse the string of model.
- gboolean gst_tensor_filter_allocate_in_invoke (GstTensorFilterPrivate *priv)

  check if the allocate_in_invoke is valid for the framework
- void gst_tensor_filter_destroy_notify_util (GstTensorFilterPrivate *priv, void *data)

  Free the data allocated for tensor filter output.
- gchar * gst_tensorsinfo_compare_to_string (const GstTensorsInfo *info1, const GstTensorsInfo *info2)

  Printout the comparison results of two tensors as a string.
- void gst_tensorsinfo_compare_print (const GstTensorsInfo *info1, const GstTensorsInfo *info2)

  Printout the comparison results of two tensors.
- void gst_tensor_filter_install_properties (GObjectClass *gobject_class)

  Installs all the properties for tensor_filter.
- void gst_tensor_filter_common_init_property (GstTensorFilterPrivate *priv)

  Initialize the properties for tensor-filter.
- void gst_tensor_filter_common_free_property (GstTensorFilterPrivate *priv)

  Free the properties for tensor-filter.
- static void gst_tensor_filter_parse_accelerator (GstTensorFilterPrivate *priv, GstTensorFilterProperties *prop, const char *accelerators)

  Parse the hardware accelerators to be used for this framework.
- static gchar * _detect_framework_from_config (const gchar *extension)

  Get available framework from config.
- gchar * gst_tensor_filter_detect_framework (const gchar *const *model_files, const guint num_models, const gboolean load_conf)

Get neural network framework name from given model file. This does not guarantee the framework is available on the target device.

- static void gst_tensor_filter_get_available_framework (GstTensorFilterPrivate *priv, const char *fw_name)

  automatically selecting framework for tensor filter
- static gint _gtfc_setprop_FRAMEWORK (GstTensorFilterPrivate *priv, GstTensorFilterProperties *prop, const GValue *value)

  Handle "PROP_FRAMEWORK" for set-property.
- static gint _gtfc_setprop_MODEL (GstTensorFilterPrivate *priv, GstTensorFilterProperties *prop, const G↩ Value *value)

  Handle "PROP_MODEL" for set-property.
- static gint _gtfc_setprop_DIMENSION (GstTensorFilterPrivate *priv, const GValue *value, const gboolean is_input)

  Handle "PROP_INPUT" and "PROP_OUTPUT" for set-property.
- static gint _gtfc_setprop_TYPE (GstTensorFilterPrivate *priv, const GValue *value, const gboolean is_input)

  Handle "PROP_INPUTTYPE" and "PROP_OUTPUTTYPE" for set-property.
- static gint _gtfc_setprop_NAME (GstTensorFilterPrivate *priv, const GValue *value, const gboolean is_input)

  Handle "PROP_INPUTNAME" and "PROP_OUTPUTNAME" for set-property.
- static gint _gtfc_setprop_CUSTOM (GstTensorFilterPrivate *priv, GstTensorFilterProperties *prop, const G↩ Value *value)

  Handle "PROP_CUSTOM" for set-property.
- static gint _gtfc_setprop_LAYOUT (GstTensorFilterPrivate *priv, const GValue *value, const gboolean is_↩ input)

  Handle "PROP_INPUTLAYOUT" and "PROP_OUTPUTLAYOUT" for set-property.
- static gint _gtfc_setprop_LATENCY (GstTensorFilterPrivate *priv, GstTensorFilterProperties *prop, const GValue *value)

  Handle "PROP_LATENCY" for set-property.
- static gint _gtfc_setprop_THROUGHPUT (GstTensorFilterPrivate *priv, GstTensorFilterProperties *prop, const GValue *value)

  Handle "PROP_THROUGHPUT" for set-property.
- static gint _gtfc_setprop_INPUTCOMBINATION (GstTensorFilterPrivate *priv, GList **prop_list, const G↩ Value *value)

  Handle "PROP_INPUTCOMBINATION" for set-property.
- static gint _gtfc_setprop_OUTPUTCOMBINATION (GstTensorFilterPrivate *priv, GList **prop_list1, GList **prop_list2, const GValue *value)

  Handle "PROP_OUTPUTCOMBINATION" for set-property.
- static gint _gtfc_setprop_SHARED_TENSOR_FILTER_KEY (GstTensorFilterProperties *prop, const GValue *value)

  Handle "PROP_SHARED_TENSOR_FILTER_KEY" for set-property.
- static gint _gtfc_setprop_PROP_INVOKE_DYNAMIC (GstTensorFilterPrivate *priv, const GValue *value)

  Handle "PROP_INVOKE_DYNAMIC" for set-property.
- static gint _gtfc_setprop_SUSPEND (GstTensorFilterPrivate *priv, const GValue *value)

  Handle "PROP_SUSPEND" for set-property.
- gboolean gst_tensor_filter_common_set_property (GstTensorFilterPrivate *priv, guint prop_id, const GValue *value, GParamSpec *pspec)

  Set the properties for tensor_filter.
- static void gst_tensor_filter_property_to_string (GValue *value, GstTensorFilterPrivate *priv, guint prop_id)

  Convert GList to GValue.
- gboolean gst_tensor_filter_common_get_property (GstTensorFilterPrivate *priv, guint prop_id, GValue *value, GParamSpec *pspec)

  Get the properties for tensor_filter.
- gboolean gst_tensor_filter_common_get_combined_in_info (GstTensorFilterPrivate *priv, const GstTensorsInfo *in, GstTensorsInfo *combined)

  Configure input tensor info with combi option.

- gboolean gst_tensor_filter_common_get_combined_out_info (GstTensorFilterPrivate *priv, const GstTensorsInfo *in, const GstTensorsInfo *out, GstTensorsInfo *combined)

  Configure output tensor info with combi option.
- gboolean gst_tensor_filter_common_get_out_info (GstTensorFilterPrivate *priv, GstTensorsInfo *in, GstTensorsInfo *out)

  Get output tensor info from NN model with given input info.
- void gst_tensor_filter_load_tensor_info (GstTensorFilterPrivate *priv)

  Load tensor info from NN model. (both input and output tensor)
- void gst_tensor_filter_common_open_fw (GstTensorFilterPrivate *priv)

  Open NN framework.
- void gst_tensor_filter_common_unload_fw (GstTensorFilterPrivate *priv)

  Unload NN framework.
- void gst_tensor_filter_common_close_fw (GstTensorFilterPrivate *priv)

  Close NN framework.
- accl_hw get_accl_hw_type (const gchar *key)

  return accl_hw type from string
- const gchar * get_accl_hw_str (const accl_hw key)

  return string based on accl_hw type
- static const gchar ** add_basic_supported_accelerators (const gchar **supported_accelerators)

  Added basic accelerators (auto, default) to supported accelerators.
- static const gchar ** filter_supported_accelerators (const gchar **supported_accelerators)

  Filter accelerators based on the runtime system.
- static accl_hw parse_accl_hw_util (const gchar *accelerators, const gchar **supported_accelerators, const gchar *auto_accelerator, const gchar *default_accelerator)

  parse user given string to extract accelerator based on given regex
- static gint runtime_check_supported_accelerator (const gchar *accl)

  Check if this accelerator can be used based on the runtime system.
- accl_hw parse_accl_hw_fill (parse_accl_args accl_args)

  parse user given string to extract accelerator based on given regex filling in optional arguments
- gboolean gst_tensor_filter_check_hw_availability (const gchar *name, const accl_hw hw, const char *custom)

  Check if the given hw is supported by the framework.
- void * nnstreamer_filter_shared_model_get (void *instance, const char *key)

  Get the shared model representation that is already shared and has the same key.
- void * nnstreamer_filter_shared_model_insert_and_get (void *instance, char *key, void *interpreter)

  Insert the new shared model representation and get the value.
- int nnstreamer_filter_shared_model_remove (void *instance, const char *key, void(*free_callback)(void *))

  Remove the instance registered at the referred list of shared model table. If referred list is empty, `free_callback` is executed.
- void nnstreamer_filter_shared_model_replace (void *instance, const char *key, void *new_interpreter, void(*replace_callback)(void *, void *), void(*free_callback)(void *))

  Helper to reload interpreter for instances that has shared key. `replace_callback` is called iterating instances in referred list.

Variables

- static const gchar * regex_accl_utils [ ]
- static const gchar * regex_accl_elem_utils [ ]
- static GHashTable * shared_model_table = NULL

9.106.1 Detailed Description

Common functions for various tensor_filters.

Date

28 Aug 2019

See also

http://github.com/nnstreamer/nnstreamer

Author

Parichay Kapoor pk.kapoor@samsung.com
MyungJoo Ham myungjoo.ham@samsung.com

Bug No known bugs except for NYI items

9.106.2 Macro Definition Documentation

9.106.2.1 g_free_const

```
#define g_free_const(
              x ) g_free((void*)(long)(x))
```

Free memory.

Definition at line 87 of file tensor_filter_common.c.

9.106.2.2 g_strfreev_const

```
#define g_strfreev_const(
              x ) g_strfreev((void*)(long)(x))
```

Definition at line 88 of file tensor_filter_common.c.

9.106.2.3 REGEX_ACCL_DELIMITER

```
#define REGEX_ACCL_DELIMITER "|"
```

Definition at line 63 of file tensor_filter_common.c.

9.106.2.4 REGEX_ACCL_ELEM_DELIMITER

```
#define REGEX_ACCL_ELEM_DELIMITER "|"
```

Definition at line 57 of file tensor_filter_common.c.

9.106.2.5 REGEX_ACCL_ELEM_END

```
#define REGEX_ACCL_ELEM_END ")?"
```

Definition at line 58 of file tensor_filter_common.c.

9.106.2.6 REGEX_ACCL_ELEM_PREFIX

```
#define REGEX_ACCL_ELEM_PREFIX "(?<!!)"
```

Definition at line 55 of file tensor_filter_common.c.

9.106.2.7 REGEX_ACCL_ELEM_START

```
#define REGEX_ACCL_ELEM_START "("
```

Basic elements to form accelerator regex forming

Definition at line 54 of file tensor_filter_common.c.

9.106.2.8 REGEX_ACCL_ELEM_SUFFIX

```
#define REGEX_ACCL_ELEM_SUFFIX ""
```

Definition at line 56 of file tensor_filter_common.c.

9.106.2.9 REGEX_ACCL_END

```
#define REGEX_ACCL_END ")*[]?))"
```

Definition at line 64 of file tensor_filter_common.c.

9.106.2.10 REGEX_ACCL_PREFIX

```
#define REGEX_ACCL_PREFIX ""
```

Definition at line 61 of file tensor_filter_common.c.

9.106.2.11 REGEX_ACCL_START

```
#define REGEX_ACCL_START "(^(true)[:]?([(]?("
```

Definition at line 60 of file tensor_filter_common.c.

9.106.2.12 REGEX_ACCL_SUFFIX

```
#define REGEX_ACCL_SUFFIX ""
```

Definition at line 62 of file tensor_filter_common.c.

9.106.2.13 silent_debug_info

```
#define silent_debug_info(
              i,
              msg )
```

Value:

```
  do { \
  if (DBG) { \
    guint info_idx; \
    gchar *dim_str; \
    ml_logd (msg " total %d", (i)->num_tensors); \
    for (info_idx = 0; info_idx < (i)->num_tensors; info_idx++) { \
      GstTensorInfo *nth = gst_tensors_info_get_nth_info (i, info_idx); \
      if (nth) { \
        dim_str = gst_tensor_get_dimension_string (nth->dimension); \
        ml_logd ("[%d] type=%d dim=%s", info_idx, nth->type, dim_str); \
        g_free (dim_str); \
      } \
    } \
  } \
} while (0)
```

Definition at line 35 of file tensor_filter_common.c.

9.106.3 Function Documentation

9.106.3.1 _detect_framework_from_config()

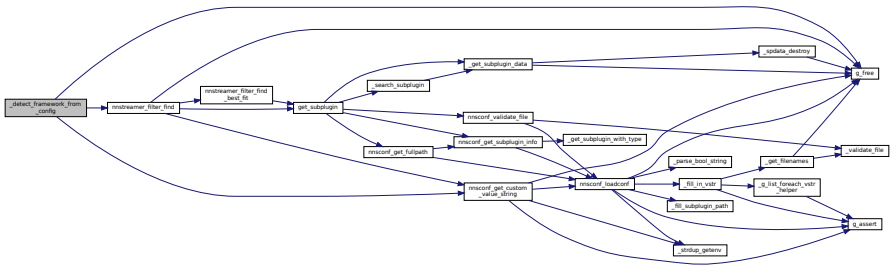
```
static gchar* _detect_framework_from_config (
            const gchar * extension )  [static]
```

Get available framework from config.

key str: framework_priority_<file ext> (e.g., for tensorflow-lite model, model_file.tflite, key str is 'framework_↩ priority_tflite'.

Definition at line 1185 of file tensor_filter_common.c.

Here is the call graph for this function:



Here is the caller graph for this function:

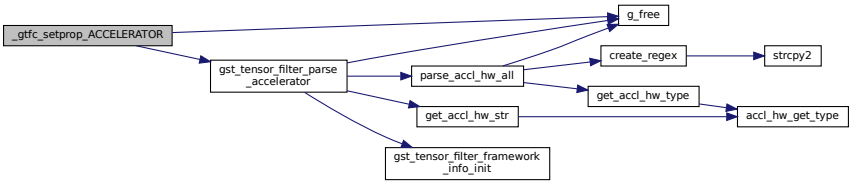
9.106.3.2 _gtfc_setprop_ACCELERATOR()

```
static gint _gtfc_setprop_ACCELERATOR (
            GstTensorFilterPrivate * priv,
            GstTensorFilterProperties * prop,
            const GValue * value )  [static]
```

Handle "PROP_ACCELERATOR" for set-property.

Definition at line 1658 of file tensor_filter_common.c.

Here is the call graph for this function:



Here is the caller graph for this function:

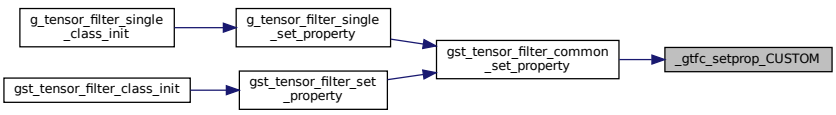
9.106.3.3 _gtfc_setprop_CUSTOM()

```
static gint _gtfc_setprop_CUSTOM (
            GstTensorFilterPrivate * priv,
            GstTensorFilterProperties * prop,
            const GValue * value )  [static]
```

Handle "PROP_CUSTOM" for set-property.

Definition at line 1627 of file tensor_filter_common.c.

Here is the caller graph for this function:



9.106.3.4 _gtfc_setprop_DIMENSION()

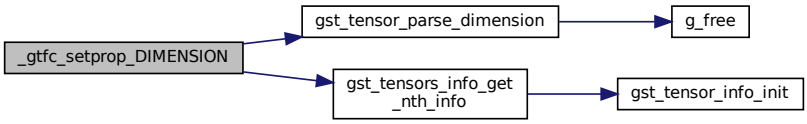
```
static gint _gtfc_setprop_DIMENSION (
            GstTensorFilterPrivate * priv,
            const GValue * value,
            const gboolean is_input )  [static]
```

Handle "PROP_INPUT" and "PROP_OUTPUT" for set-property.

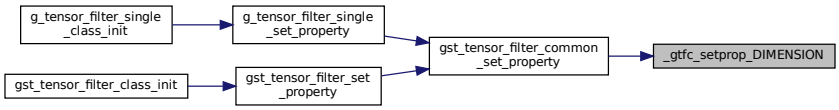
Once configured, it cannot be changed in runtime for now

Definition at line 1490 of file tensor_filter_common.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.106.3.5 _gtfc_setprop_FRAMEWORK()

```
static gint _gtfc_setprop_FRAMEWORK (
            GstTensorFilterPrivate * priv,
            GstTensorFilterProperties * prop,
            const GValue * value )  [static]
```

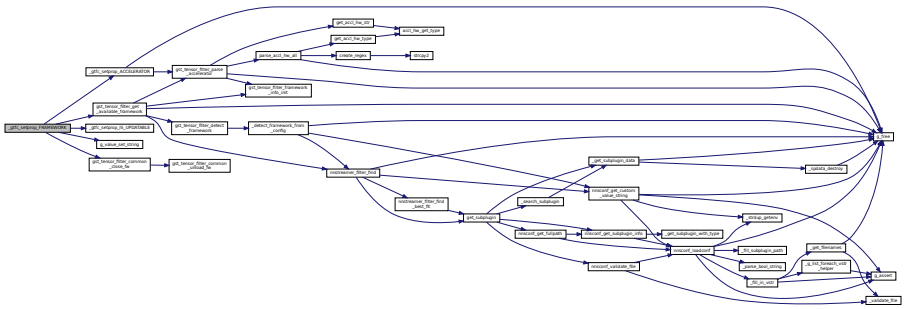
Handle "PROP_FRAMEWORK" for set-property.

set PROP_IS_UPDATABLE in case it was set before framework

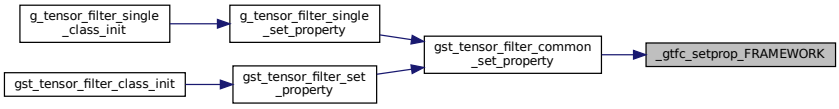
set PROP_ACCELERATOR in case it was set before framework

Definition at line 1381 of file tensor_filter_common.c.

Here is the call graph for this function:



Here is the caller graph for this function:



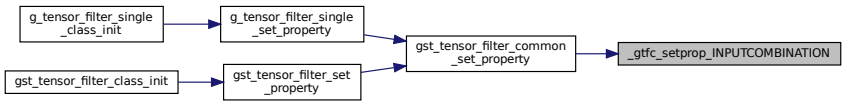
9.106.3.6 _gtfc_setprop_INPUTCOMBINATION()

```
static gint _gtfc_setprop_INPUTCOMBINATION (
            GstTensorFilterPrivate * priv,
            GList ** prop_list,
            const GValue * value )  [static]
```

Handle "PROP_INPUTCOMBINATION" for set-property.

Definition at line 1843 of file tensor_filter_common.c.

Here is the caller graph for this function:



9.106.3.7 _gtfc_setprop_IS_UPDATABLE()

```
static gint _gtfc_setprop_IS_UPDATABLE (
            GstTensorFilterPrivate * priv,
            GstTensorFilterProperties * prop,
            const GValue * value )  [static]
```

Handle "PROP_IS_UPDATABLE" for set-property.

Definition at line 1710 of file tensor_filter_common.c.

Here is the caller graph for this function:

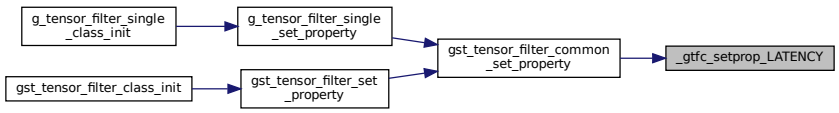
9.106.3.8 _gtfc_setprop_LATENCY()

```
static gint _gtfc_setprop_LATENCY (
            GstTensorFilterPrivate * priv,
            GstTensorFilterProperties * prop,
            const GValue * value )  [static]
```

Handle "PROP_LATENCY" for set-property.

Definition at line 1799 of file tensor_filter_common.c.

Here is the caller graph for this function:



9.106.3.9 _gtfc_setprop_LAYOUT()

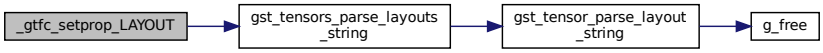
```
static gint _gtfc_setprop_LAYOUT (
            GstTensorFilterPrivate * priv,
            const GValue * value,
            const gboolean is_input )  [static]
```

Handle "PROP_INPUTLAYOUT" and "PROP_OUTPUTLAYOUT" for set-property.

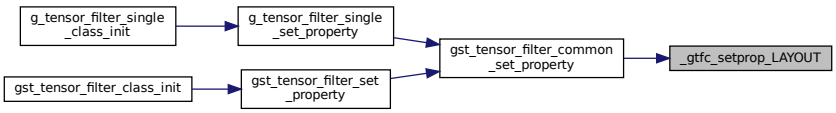
Update the properties

Definition at line 1732 of file tensor_filter_common.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.106.3.10 _gtfc_setprop_MODEL()

```
static gint _gtfc_setprop_MODEL (
            GstTensorFilterPrivate * priv,
            GstTensorFilterProperties * prop,
            const GValue * value )  [static]
```

Handle "PROP_MODEL" for set-property.

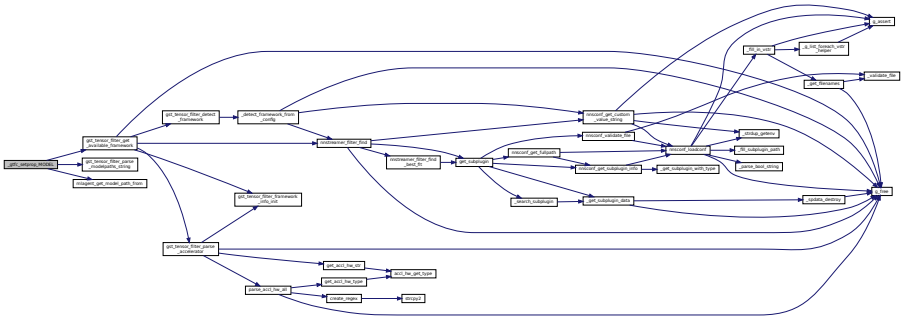
Store a copy of the original prop in case the reload fails

Reload model if FW has been already opened; In the case of reloading model files, each priv->fw (tensor filter for each nnfw) has responsibility for the verification of the path regardless of priv->fw->verify_model_path.

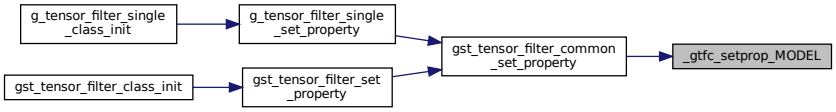
original prop is sent and not the updated prop

Definition at line 1428 of file tensor_filter_common.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.106.3.11 _gtfc_setprop_NAME()

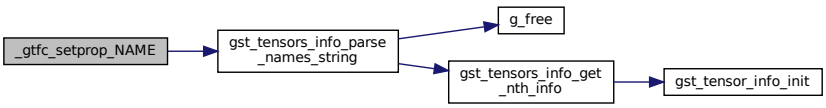
```
static gint _gtfc_setprop_NAME (
            GstTensorFilterPrivate * priv,
            const GValue * value,
            const gboolean is_input )  [static]
```

Handle "PROP_INPUTNAME" and "PROP_OUTPUTNAME" for set-property.

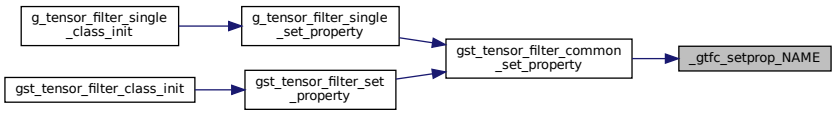
Once configured, it cannot be changed in runtime for now

Definition at line 1588 of file tensor_filter_common.c.

Here is the call graph for this function:



Here is the caller graph for this function:



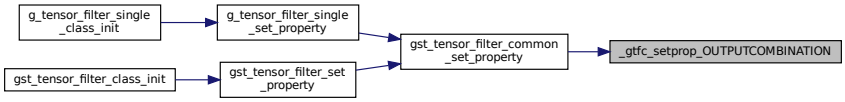
9.106.3.12 _gtfc_setprop_OUTPUTCOMBINATION()

```
static gint _gtfc_setprop_OUTPUTCOMBINATION (
            GstTensorFilterPrivate * priv,
            GList ** prop_list1,
            GList ** prop_list2,
            const GValue * value )  [static]
```

Handle "PROP_OUTPUTCOMBINATION" for set-property.

Definition at line 1874 of file tensor_filter_common.c.

Here is the caller graph for this function:



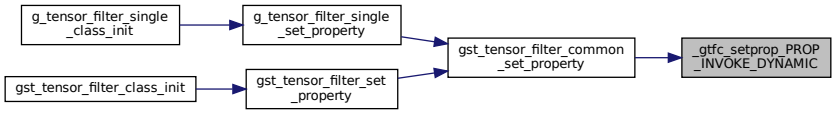
9.106.3.13 _gtfc_setprop_PROP_INVOKE_DYNAMIC()

```
static gint _gtfc_setprop_PROP_INVOKE_DYNAMIC (
            GstTensorFilterPrivate * priv,
            const GValue * value )  [static]
```

Handle "PROP_INVOKE_DYNAMIC" for set-property.

Definition at line 1937 of file tensor_filter_common.c.

Here is the caller graph for this function:



9.106.3.14 _gtfc_setprop_SHARED_TENSOR_FILTER_KEY()

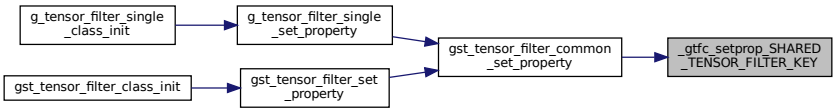
```
static gint _gtfc_setprop_SHARED_TENSOR_FILTER_KEY (
            GstTensorFilterProperties * prop,
            const GValue * value )  [static]
```

Handle "PROP_SHARED_TENSOR_FILTER_KEY" for set-property.

Definition at line 1917 of file tensor_filter_common.c.

Here is the call graph for this function:

Here is the caller graph for this function:



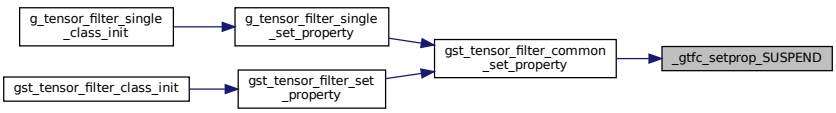
9.106.3.15 _gtfc_setprop_SUSPEND()

```
static gint _gtfc_setprop_SUSPEND (
            GstTensorFilterPrivate * priv,
            const GValue * value )  [static]
```

Handle "PROP_SUSPEND" for set-property.

Definition at line 1950 of file tensor_filter_common.c.

Here is the caller graph for this function:



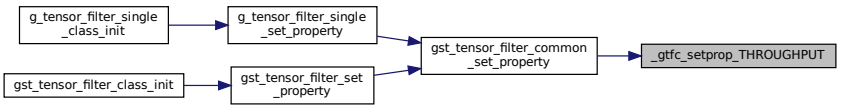
9.106.3.16 _gtfc_setprop_THROUGHPUT()

```
static gint _gtfc_setprop_THROUGHPUT (
            GstTensorFilterPrivate * priv,
            GstTensorFilterProperties * prop,
            const GValue * value )  [static]
```

Handle "PROP_THROUGHPUT" for set-property.

Definition at line 1821 of file tensor_filter_common.c.

Here is the caller graph for this function:



9.106.3.17 _gtfc_setprop_TYPE()

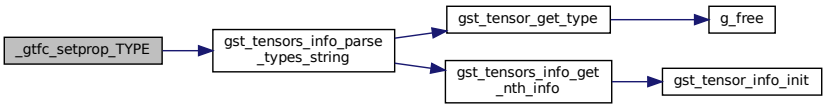
```
static gint _gtfc_setprop_TYPE (
            GstTensorFilterPrivate * priv,
            const GValue * value,
            const gboolean is_input )  [static]
```

Handle "PROP_INPUTTYPE" and "PROP_OUTPUTTYPE" for set-property.

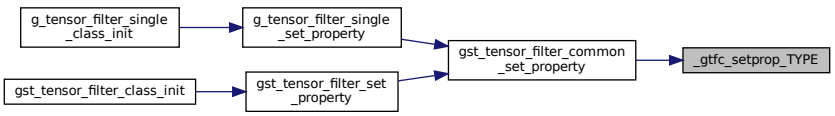
Once configured, it cannot be changed in runtime for now

Definition at line 1549 of file tensor_filter_common.c.

Here is the call graph for this function:



Here is the caller graph for this function:



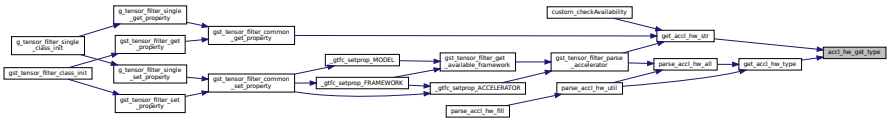
9.106.3.18 accl_hw_get_type()

```
static GType accl_hw_get_type (
            void  )  [static]
```

to get and register hardware accelerator backend enum

Definition at line 2883 of file tensor_filter_common.c.

Here is the caller graph for this function:



9.106.3.19 add_basic_supported_accelerators()

```
static const gchar** add_basic_supported_accelerators (
            const gchar ** supported_accelerators )  [static]
```

Added basic accelerators (auto, default) to supported accelerators.

Note

returned array must be freed by the caller

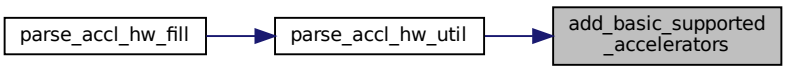
Count number of elements for the array

Allocate the array

Fill the array

Definition at line 2686 of file tensor_filter_common.c.

Here is the caller graph for this function:



9.106.3.20 create_regex()

```
static gchar* create_regex (
            const gchar ** enum_list,
            const gchar ** regex_utils )  [static]
```

create regex for the given string list and regex basic elements

Parameters

in	*enum_list*	list of strings to form regex for
in	*regex_utils*	list of basic elements to form regex

Returns

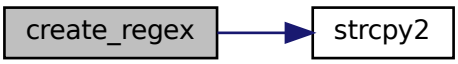
the formed regex (to be freed by the caller), NULL on error

create the regex string

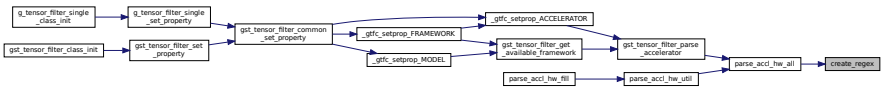
escape the special characters

Definition at line 431 of file tensor_filter_common.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.106.3.21 filter_supported_accelerators()

```
static const gchar** filter_supported_accelerators (
            const gchar ** supported_accelerators )  [static]
```

Filter accelerators based on the runtime system.

Note

returned array must be freed by the caller

This filters out NEON accelerator if the system running the tensor_filter does not support NEON instructions Count number of elements for the array

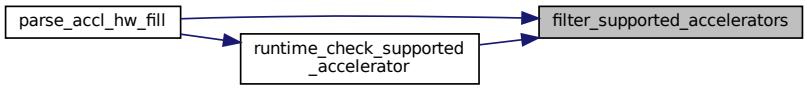
Allocate the array

Fill the array

Definition at line 2722 of file tensor_filter_common.c.

Here is the call graph for this function:

Here is the caller graph for this function:



9.106.3.22 G_LOCK_DEFINE_STATIC()

```
G_LOCK_DEFINE_STATIC (
            shared_model_table  )
```

mutex for shared model table.

9.106.3.23 get_accl_hw_str()

```
const gchar* get_accl_hw_str (
            const accl_hw key )
```

return string based on accl_hw type

Parameters

key	The key enum value

Returns

Corresponding string. Returns ACCL_NONE_STR if not found.

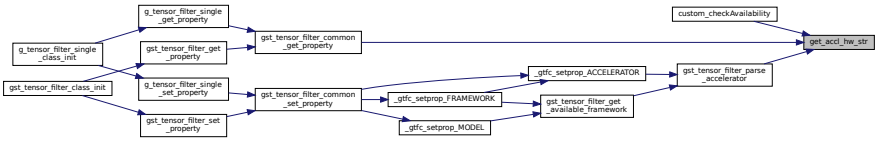
Note

Do not free the returned char *

Definition at line 2598 of file tensor_filter_common.c.

Here is the call graph for this function:

Here is the caller graph for this function:



9.106.3.24 get_accl_hw_type()

```
accl_hw get_accl_hw_type (
            const gchar * key )
```

return accl_hw type from string

Parameters

key	The key string value

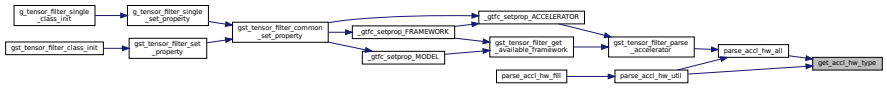
Returns

Corresponding index. Returns ACCL_NONE if not found.

Definition at line 2577 of file tensor_filter_common.c.

Here is the call graph for this function:

Here is the caller graph for this function:



9.106.3.25 gst_tensor_filter_allocate_in_invoke()

```
gboolean gst_tensor_filter_allocate_in_invoke (
            GstTensorFilterPrivate * priv )
```

check if the allocate_in_invoke is valid for the framework

Parameters

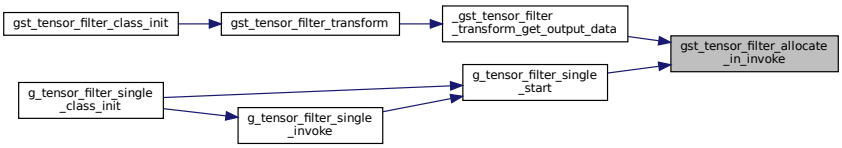
in	*priv*	Struct containing the properties of the object

Returns

TRUE if valid, FALSE on error

Definition at line 757 of file tensor_filter_common.c.

Here is the caller graph for this function:



9.106.3.26 gst_tensor_filter_check_hw_availability()

```
gboolean gst_tensor_filter_check_hw_availability (
            const gchar * name,
            const accl_hw hw,
            const char * custom )
```

Check if the given hw is supported by the framework.

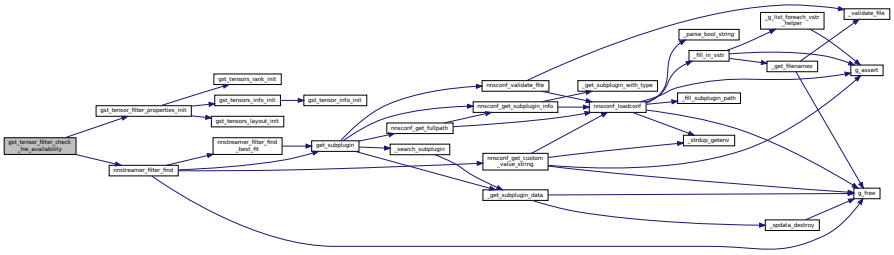
Note

This function is included in nnstreamer internal header for native APIs. When changing the declaration, you should update the internal header (nnstreamer_internal.h).

Only check for specific HW, DEFAULT/AUTO are always supported

Definition at line 2923 of file tensor_filter_common.c.

Here is the call graph for this function:



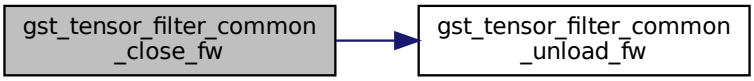
9.106.3.27 gst_tensor_filter_common_close_fw()

```
void gst_tensor_filter_common_close_fw (
            GstTensorFilterPrivate * priv )
```

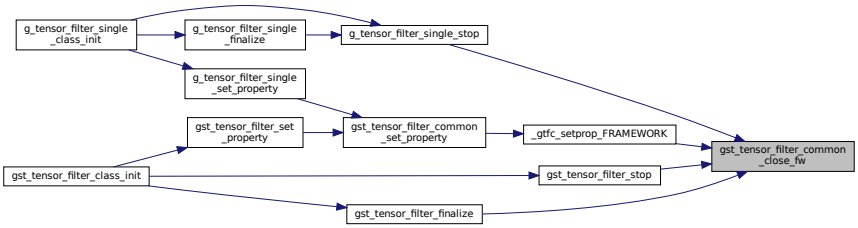
Close NN framework.

Definition at line 2560 of file tensor_filter_common.c.

Here is the call graph for this function:



Here is the caller graph for this function:



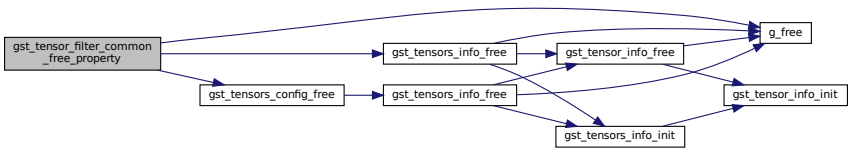
9.106.3.28 gst_tensor_filter_common_free_property()

```
void gst_tensor_filter_common_free_property (
            GstTensorFilterPrivate * priv )
```

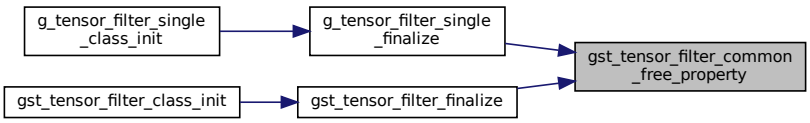
Free the properties for tensor-filter.

Definition at line 1065 of file tensor_filter_common.c.

Here is the call graph for this function:



Here is the caller graph for this function:



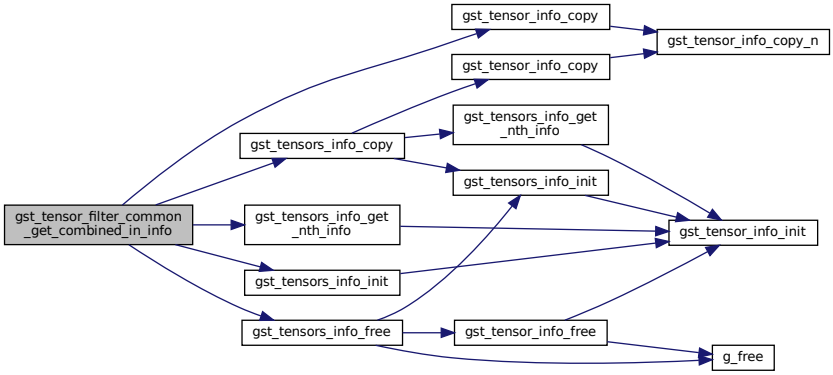
9.106.3.29 gst_tensor_filter_common_get_combined_in_info()

```
gboolean gst_tensor_filter_common_get_combined_in_info (
            GstTensorFilterPrivate * priv,
            const GstTensorsInfo * in,
            GstTensorsInfo * combined )
```

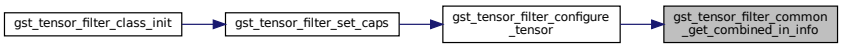
Configure input tensor info with combi option.

Definition at line 2270 of file tensor_filter_common.c.

Here is the call graph for this function:



Here is the caller graph for this function:



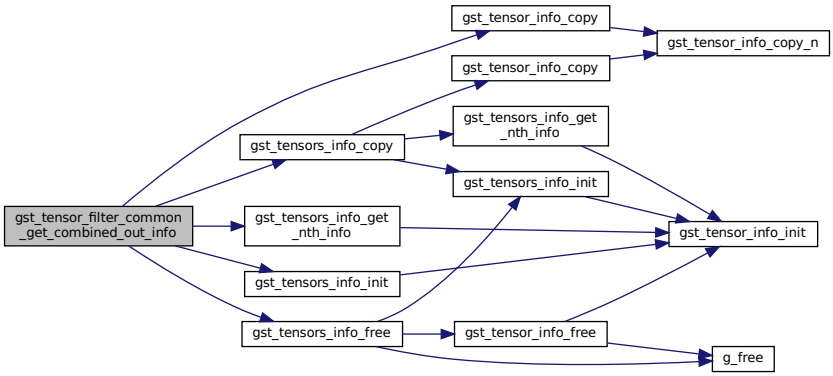
9.106.3.30 gst_tensor_filter_common_get_combined_out_info()

```
gboolean gst_tensor_filter_common_get_combined_out_info (
            GstTensorFilterPrivate * priv,
            const GstTensorsInfo * in,
            const GstTensorsInfo * out,
            GstTensorsInfo * combined )
```

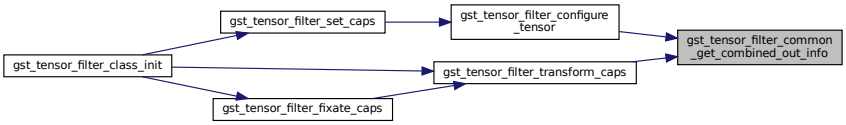
Configure output tensor info with combi option.

Definition at line 2315 of file tensor_filter_common.c.

Here is the call graph for this function:



Here is the caller graph for this function:



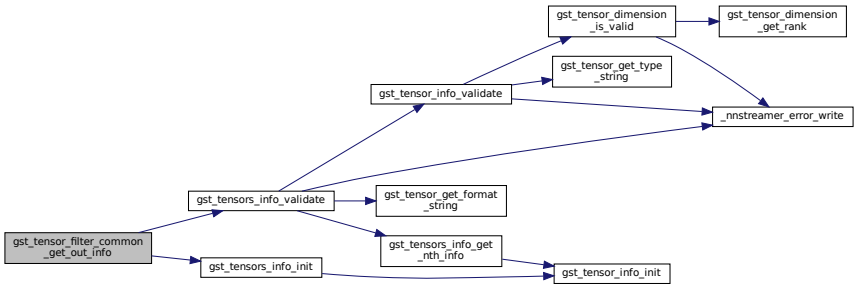
9.106.3.31 gst_tensor_filter_common_get_out_info()

```
gboolean gst_tensor_filter_common_get_out_info (
            GstTensorFilterPrivate * priv,
            GstTensorsInfo * in,
            GstTensorsInfo * out )
```

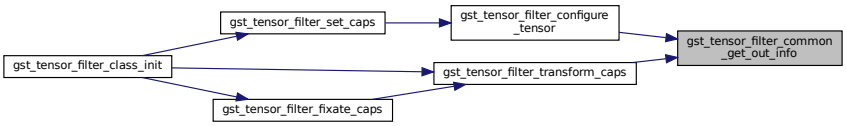
Get output tensor info from NN model with given input info.

Definition at line 2374 of file tensor_filter_common.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.106.3.32 gst_tensor_filter_common_get_property()

```
gboolean gst_tensor_filter_common_get_property (
            GstTensorFilterPrivate * priv,
            guint prop_id,
            GValue * value,
            GParamSpec * pspec )
```

Get the properties for tensor_filter.

Parameters

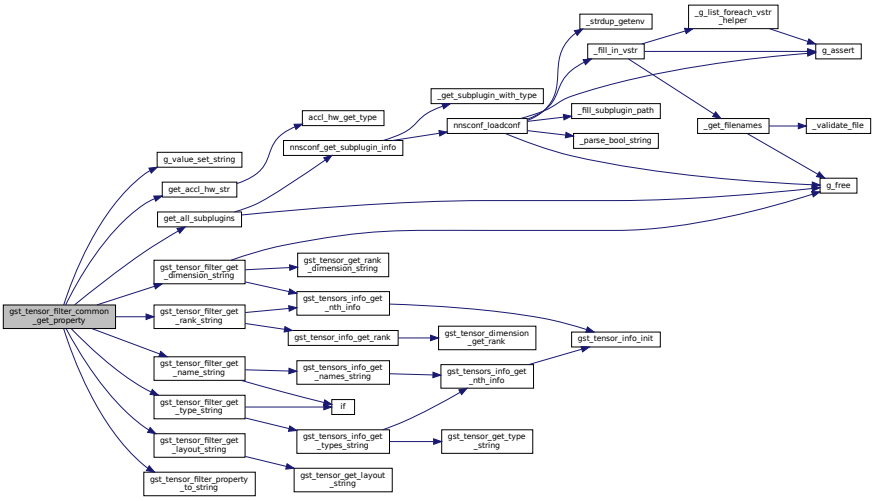
in	*priv*	Struct containing the properties of the object
in	*prop_id*	Id for the property
in	*value*	Container to return the asked property
in	*pspec*	Metadata to specify the parameter

Returns

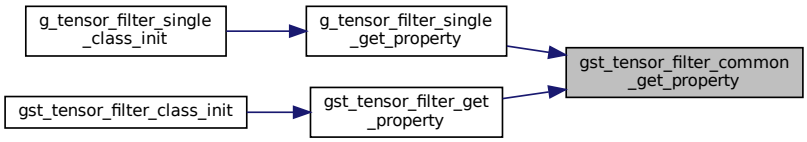
TRUE if prop_id is value, else FALSE

Definition at line 2102 of file tensor_filter_common.c.

Here is the call graph for this function:



Here is the caller graph for this function:



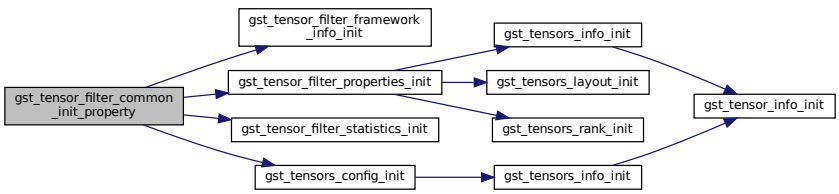
9.106.3.33 gst_tensor_filter_common_init_property()

```
void gst_tensor_filter_common_init_property (
            GstTensorFilterPrivate * priv )
```

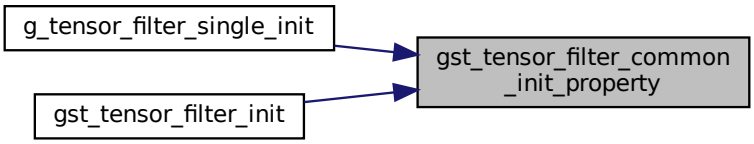
Initialize the properties for tensor-filter.

Definition at line 1041 of file tensor_filter_common.c.

Here is the call graph for this function:



Here is the caller graph for this function:



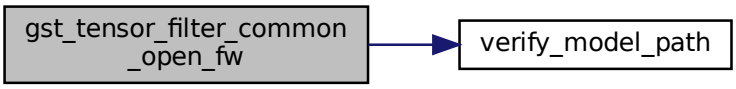
9.106.3.34 gst_tensor_filter_common_open_fw()

```
void gst_tensor_filter_common_open_fw (
            GstTensorFilterPrivate * priv )
```

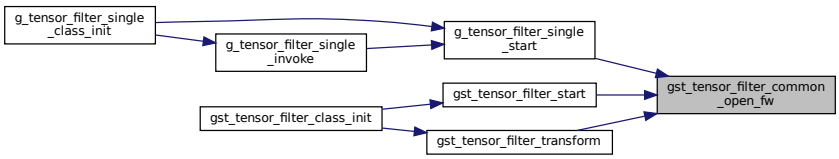
Open NN framework.

Definition at line 2491 of file tensor_filter_common.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.106.3.35 gst_tensor_filter_common_set_property()

```
gboolean gst_tensor_filter_common_set_property (
            GstTensorFilterPrivate * priv,
            guint prop_id,
            const GValue * value,
            GParamSpec * pspec )
```

Set the properties for tensor_filter.

Parameters

in	*priv*	Struct containing the properties of the object
in	*prop_id*	Id for the property
in	*value*	Container to return the asked property
in	*pspec*	Metadata to specify the parameter

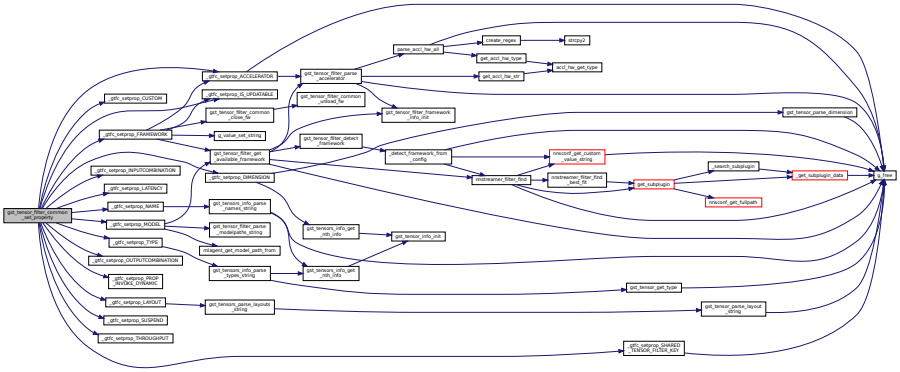
Returns

TRUE if prop_id is value, else FALSE

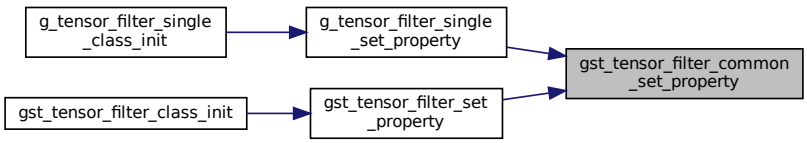
Although no one return !0 at this point, let's enable error handling.

Definition at line 1967 of file tensor_filter_common.c.

Here is the call graph for this function:



Here is the caller graph for this function:



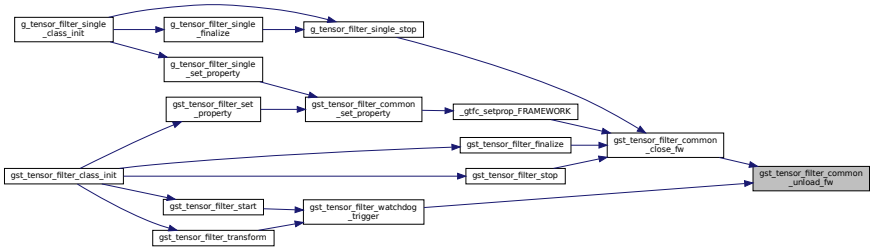
9.106.3.36 gst_tensor_filter_common_unload_fw()

```
void gst_tensor_filter_common_unload_fw (
            GstTensorFilterPrivate * priv )
```

Unload NN framework.

Definition at line 2545 of file tensor_filter_common.c.

Here is the caller graph for this function:



9.106.3.37 gst_tensor_filter_destroy_notify_util()

```
void gst_tensor_filter_destroy_notify_util (
            GstTensorFilterPrivate * priv,
            void * data )
```

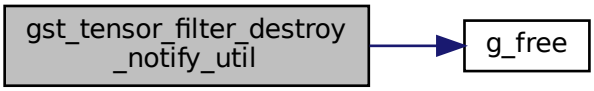
Free the data allocated for tensor filter output.

Parameters

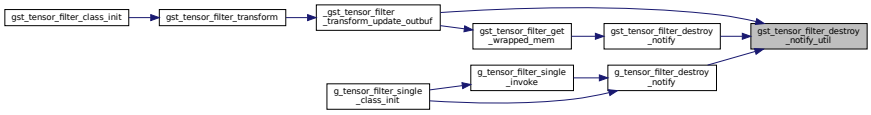
in	*priv*	Struct containing the properties of the object
in	*data*	Data to be freed

Definition at line 786 of file tensor_filter_common.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.106.3.38 gst_tensor_filter_detect_framework()

```
gchar* gst_tensor_filter_detect_framework (
            const gchar *const * model_files,
            const guint num_models,
            const gboolean load_conf )
```

Get neural network framework name from given model file. This does not guarantee the framework is available on the target device.

Parameters

in	*model_files*	the prediction model paths
in	*num_models*	the number of model files
in	*load_conf*	flag to load configuration for the priority of framework

Returns

Possible framework name (NULL if it fails to detect automatically). Caller should free returned value using g_free().

Note

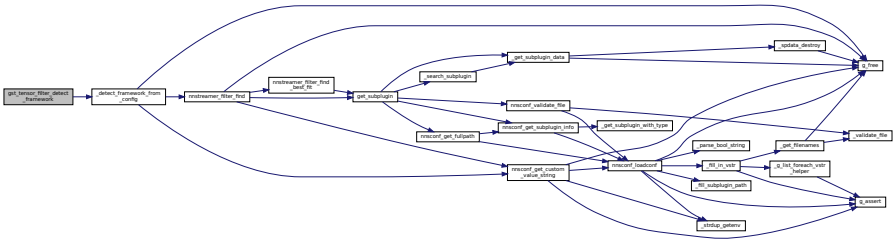
This function is included in nnstreamer internal header for native APIs. When changing the declaration, you should update the internal header (nnstreamer_internal.h).

Note

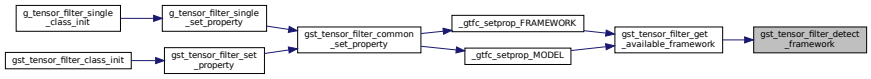
When adding new file extension for auto fw detection, you should check the condition to validate model file - ml_validate_model_file() in ML-API.

Definition at line 1232 of file tensor_filter_common.c.

Here is the call graph for this function:



Here is the caller graph for this function:



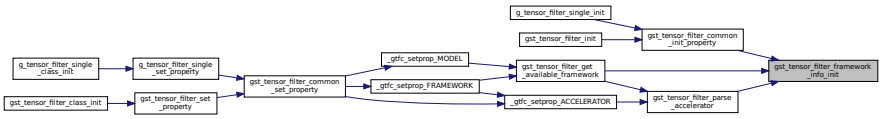
9.106.3.39 gst_tensor_filter_framework_info_init()

```
static void gst_tensor_filter_framework_info_init (
            GstTensorFilterFrameworkInfo * info )  [static]
```

Initialize the GstTensorFilterFrameworkInfo object.

Definition at line 528 of file tensor_filter_common.c.

Here is the caller graph for this function:



9.106.3.40 gst_tensor_filter_get_available_framework()

```
static void gst_tensor_filter_get_available_framework (
            GstTensorFilterPrivate * priv,
            const char * fw_name )  [static]
```

automatically selecting framework for tensor filter

Parameters

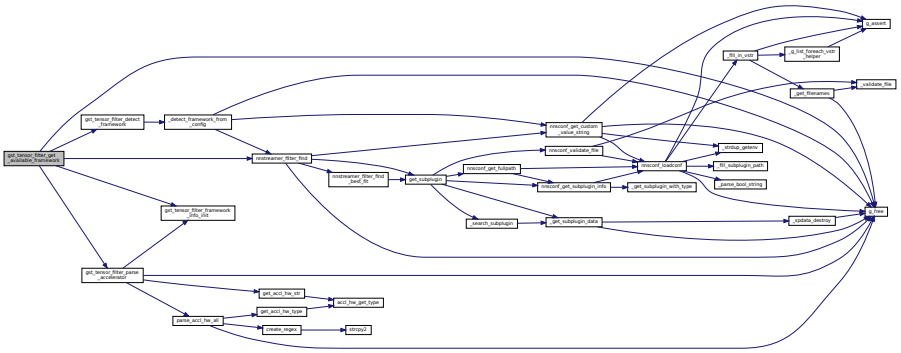
in	*priv*	Struct containing the properties of the object
in	*fw_name*	Framework name

Get framework info for v1

update the accelerator if already set based on v0 or v1

Definition at line 1317 of file tensor_filter_common.c.

Here is the call graph for this function:



Here is the caller graph for this function:

9.106.3.41 gst_tensor_filter_get_dimension_string()

```
static gchar* gst_tensor_filter_get_dimension_string (
            const GstTensorFilterProperties * prop,
            const gboolean isInput )  [static]
```

Get the dimension string of tensors considering rank count.

Parameters

in	*prop*	GstTensorFilterProperties object
	isInput	TRUE if target is input tensors

Returns

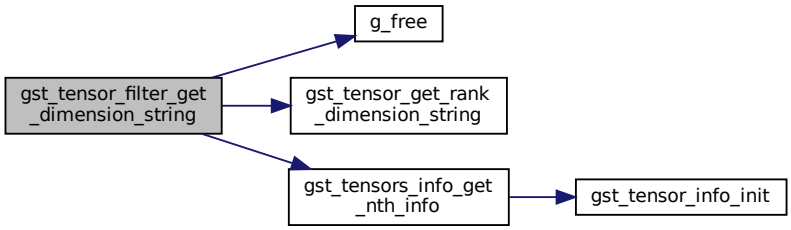
dimension string of tensors

Note

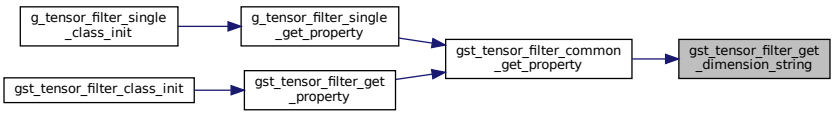
If rank count is 3, then returned string is 'd1:d2:d3`.

Definition at line 259 of file tensor_filter_common.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.106.3.42 gst_tensor_filter_get_layout_string()

```
static gchar* gst_tensor_filter_get_layout_string (
            const GstTensorFilterProperties * prop,
            const gboolean is_input )  [static]
```

Get the string of layout of tensors.

Parameters

layout	layout of the tensors

Returns

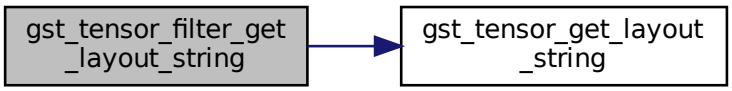
string of layouts in tensors

Note

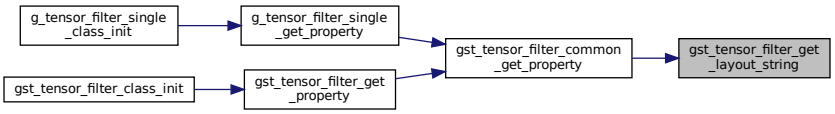
The returned value should be freed with g_free()

Definition at line 372 of file tensor_filter_common.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.106.3.43 gst_tensor_filter_get_name_string()

```
static gchar* gst_tensor_filter_get_name_string (
            const GstTensorFilterProperties * prop,
            const gboolean is_input )  [static]
```

Get the name string of tensors.

Parameters

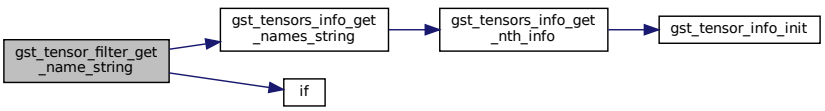
in	*prop*	GstTensorFilterProperties object
	is_input	TRUE if target is input tensors

Returns

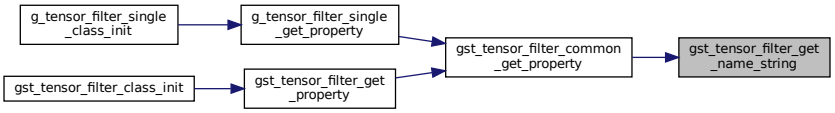
name string of tensors

Definition at line 327 of file tensor_filter_common.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.106.3.44 gst_tensor_filter_get_rank_string()

```
static gchar* gst_tensor_filter_get_rank_string (
            const GstTensorFilterProperties * prop,
            gboolean isInput )  [static]
```

Get the rank string of the tensors.

Parameters

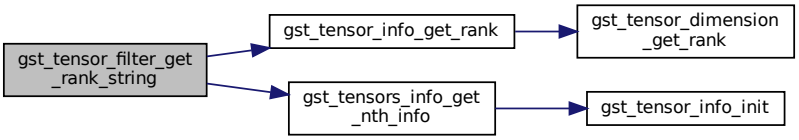
prop	GstTensorFilterProperties object
isInput	TRUE if target is input tensors

Returns

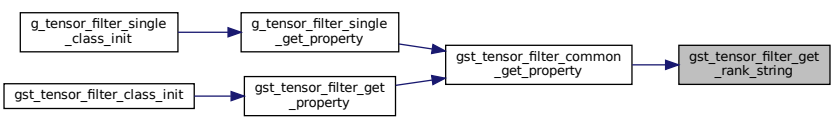
rank string of the tensors

Definition at line 211 of file tensor_filter_common.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.106.3.45 gst_tensor_filter_get_type_string()

```
static gchar* gst_tensor_filter_get_type_string (
            const GstTensorFilterProperties * prop,
            const gboolean is_input )  [static]
```

Get the type string of tensors.

Parameters

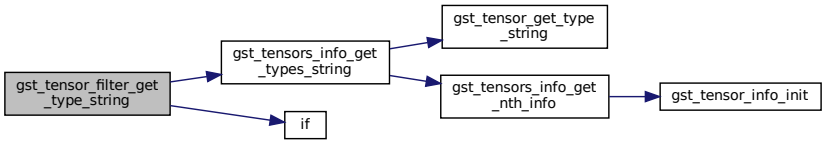
in	*prop*	GstTensorFilterProperties object
	is_input	TRUE if target is input tensors

Returns

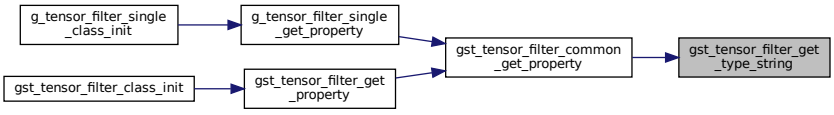
type string of tensors

Definition at line 304 of file tensor_filter_common.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.106.3.46 gst_tensor_filter_install_properties()

```
void gst_tensor_filter_install_properties (
            GObjectClass * gobject_class )
```

Installs all the properties for tensor_filter.

Parameters

in	*gobject_class*	Glib object class whose properties will be set

min

max

default: off

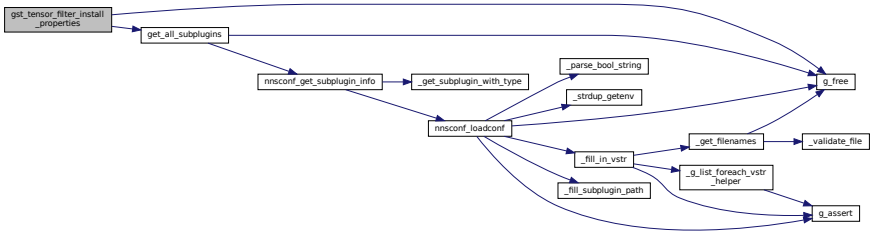
min

max

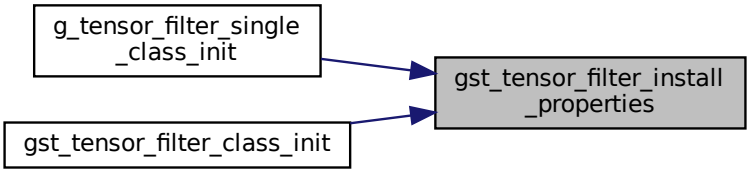
default: off

Definition at line 888 of file tensor_filter_common.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.106.3.47 gst_tensor_filter_load_tensor_info()

```
void gst_tensor_filter_load_tensor_info (
            GstTensorFilterPrivate * priv )
```

Load tensor info from NN model. (both input and output tensor)

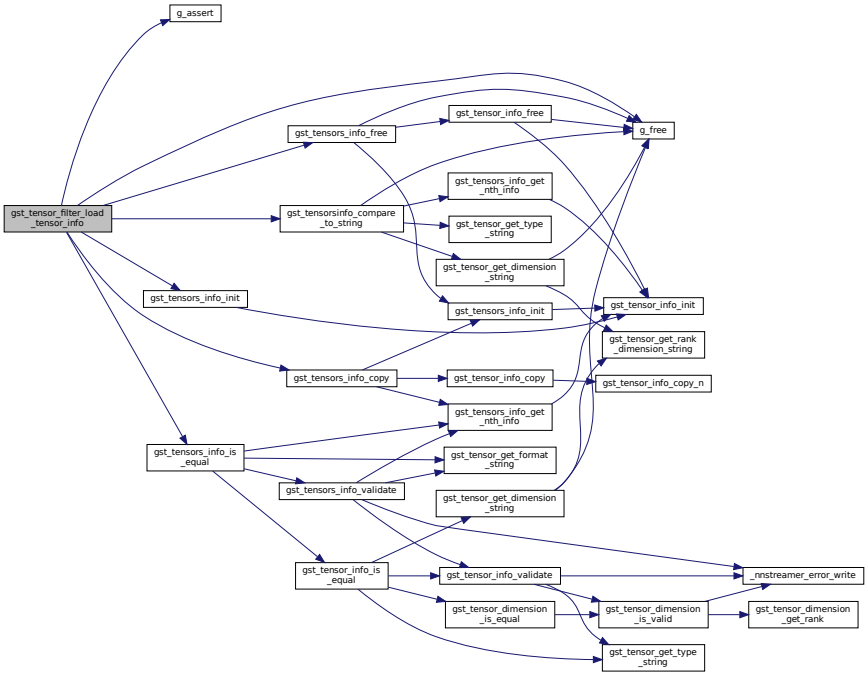
if set-property called and already has info, verify it!

In case of dynamic invoke, output tensors info is determined after invoke.

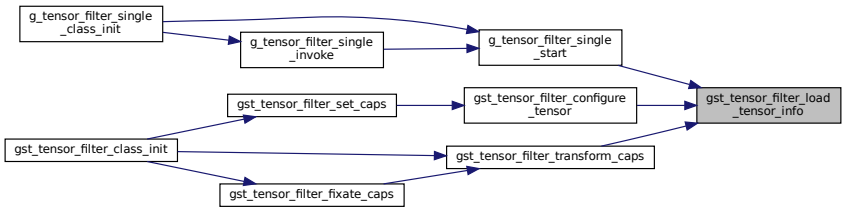
if set-property called and already has info, verify it!

Definition at line 2409 of file tensor_filter_common.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.106.3.48 gst_tensor_filter_parse_accelerator()

```
static void gst_tensor_filter_parse_accelerator (
            GstTensorFilterPrivate * priv,
            GstTensorFilterProperties * prop,
            const char * accelerators )  [static]
```

Parse the hardware accelerators to be used for this framework.

Parameters

in	*priv*	Struct containing the properties of the object
in	*prop*	Struct containing the properties of the framework
in	*accelerators*	user given input for hardware accelerators

Note

The order of preference set by the user is maintained

Get h/w accelerators supported by framework (V1 only)

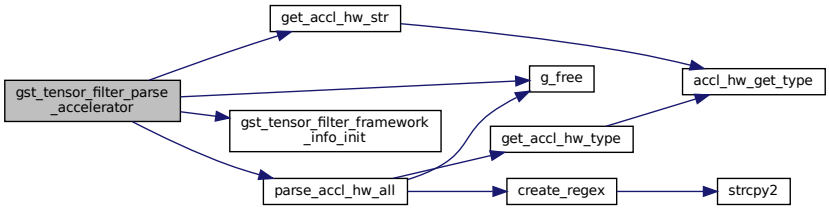
Convert the list to string format Extra 2 entries for basic accelerators : auto and default

Parse the user given h/w accelerators intersected with supported h/w

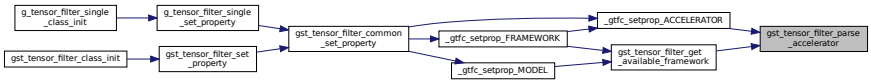
Convert the GList to regular array

Definition at line 1123 of file tensor_filter_common.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.106.3.49 gst_tensor_filter_parse_modelpaths_string()

```
static void gst_tensor_filter_parse_modelpaths_string (
            GstTensorFilterProperties * prop,
            const gchar * model_files )  [static]
```

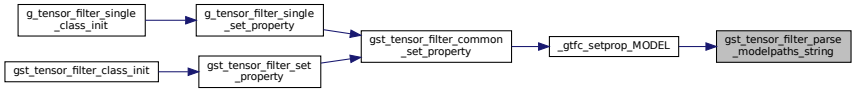
Parse the string of model.

Parameters

out	*prop*	Struct containing the properties of the object
in	*model_files*	the prediction model paths

Definition at line 728 of file tensor_filter_common.c.

Here is the caller graph for this function:



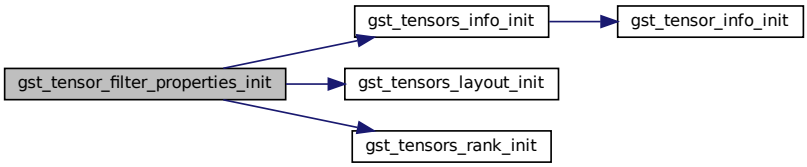
9.106.3.50 gst_tensor_filter_properties_init()

```
static void gst_tensor_filter_properties_init (
            GstTensorFilterProperties * prop )  [static]
```

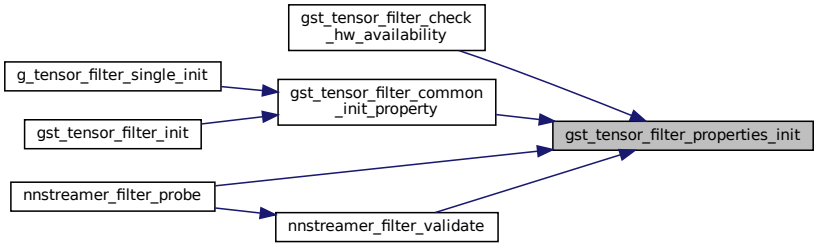
Initialize the GstTensorFilterProperties object.

Definition at line 510 of file tensor_filter_common.c.

Here is the call graph for this function:



Here is the caller graph for this function:



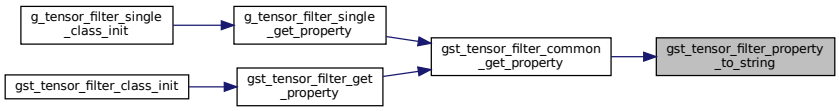
9.106.3.51 gst_tensor_filter_property_to_string()

```
static void gst_tensor_filter_property_to_string (
            GValue * value,
            GstTensorFilterPrivate * priv,
            guint prop_id )  [static]
```

Convert GList to GValue.

Definition at line 2063 of file tensor_filter_common.c.

Here is the caller graph for this function:



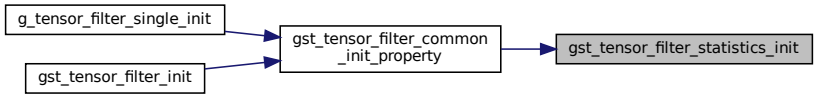
9.106.3.52 gst_tensor_filter_statistics_init()

```
static void gst_tensor_filter_statistics_init (
            GstTensorFilterStatistics * stat )  [static]
```

Initialize the GstTensorFilterFrameworkInfo object.

Definition at line 545 of file tensor_filter_common.c.

Here is the caller graph for this function:



9.106.3.53 gst_tensor_get_layout_string()

```
static const gchar* gst_tensor_get_layout_string (
            tensor_layout layout )  [static]
```

Get layout string of tensor layout.

Parameters

layout	layout of the tensor

Returns

string of layout in tensor

Definition at line 349 of file tensor_filter_common.c.

Here is the caller graph for this function:

9.106.3.54 gst_tensor_parse_layout_string()

```
static tensor_layout gst_tensor_parse_layout_string (
            const gchar * layoutstr )  [static]
```

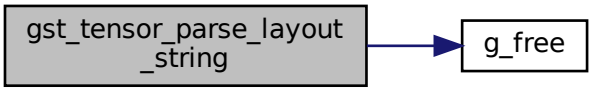
Get tensor layout from string input.

Returns

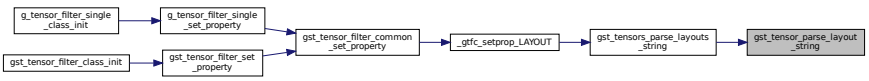
Corresponding tensor_layout.

Definition at line 134 of file tensor_filter_common.c.

Here is the call graph for this function:



Here is the caller graph for this function:



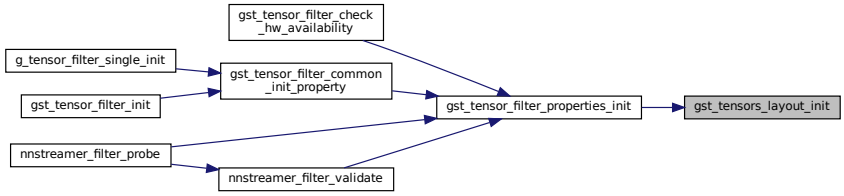
9.106.3.55 gst_tensors_layout_init()

```
static void gst_tensors_layout_init (
            tensors_layout layout )  [static]
```

Initialize the tensors layout.

Definition at line 108 of file tensor_filter_common.c.

Here is the caller graph for this function:



9.106.3.56 gst_tensors_parse_layouts_string()

```
static guint gst_tensors_parse_layouts_string (
            tensors_layout layout,
            const gchar * layout_string )  [static]
```

Parse the string of tensor layouts.

Parameters

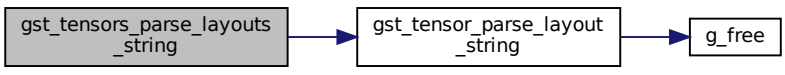
layout	layout of the tensors
layout_string	string of layout

Returns

number of parsed layouts

Definition at line 173 of file tensor_filter_common.c.

Here is the call graph for this function:



Here is the caller graph for this function:

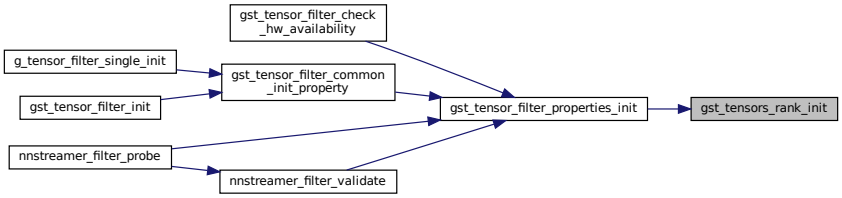
9.106.3.57 gst_tensors_rank_init()

```
static void gst_tensors_rank_init (
            unsigned int ranks[] )  [static]
```

Initialize the tensors ranks.

Definition at line 121 of file tensor_filter_common.c.

Here is the caller graph for this function:



9.106.3.58 gst_tensorsinfo_compare_print()

```
void gst_tensorsinfo_compare_print (
            const GstTensorsInfo * info1,
            const GstTensorsInfo * info2 )
```

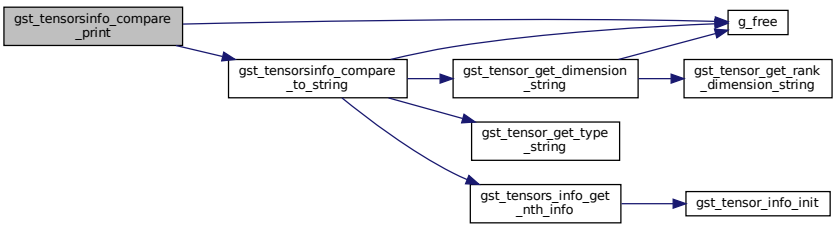
Printout the comparison results of two tensors.

Parameters

in	*info1*	The tensors to be shown on the left hand side
in	*info2*	The tensors to be shown on the right hand side

Definition at line 874 of file tensor_filter_common.c.

Here is the call graph for this function:



9.106.3.59 gst_tensorsinfo_compare_to_string()

```
gchar* gst_tensorsinfo_compare_to_string (
            const GstTensorsInfo * info1,
            const GstTensorsInfo * info2 )
```

Printout the comparison results of two tensors as a string.

Parameters

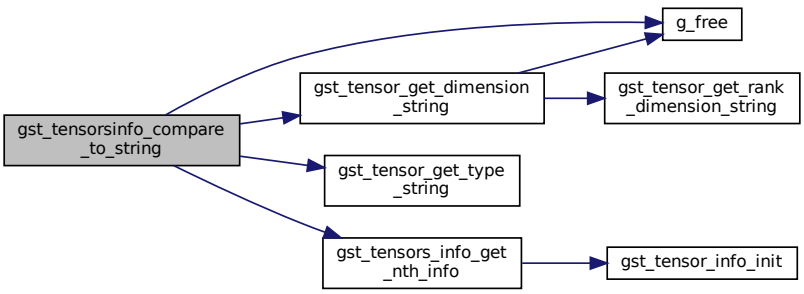
in	*info1*	The tensors to be shown on the left hand side
in	*info2*	The tensors to be shown on the right hand side

Returns

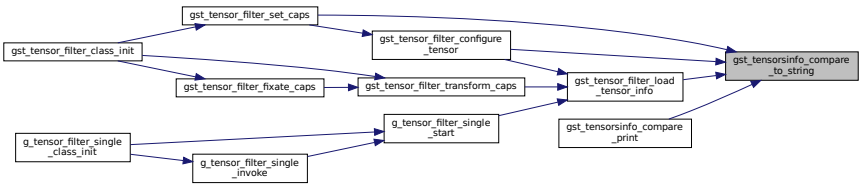
The printout string allocated. Caller should free the value.

Definition at line 811 of file tensor_filter_common.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.106.3.60 nnstreamer_filter_exit()

```
void nnstreamer_filter_exit (
            const char * name )
```

Filter's sub-plugin may call this to unregister itself.

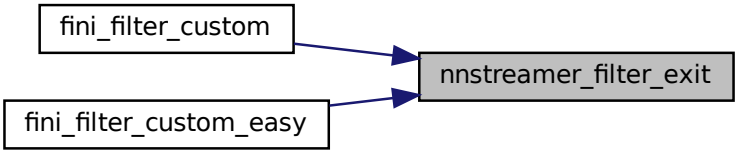
Parameters

in	*name*	The name of filter sub-plugin.

Definition at line 638 of file tensor_filter_common.c.

Here is the call graph for this function:

Here is the caller graph for this function:



9.106.3.61 nnstreamer_filter_find()

```
const GstTensorFilterFramework* nnstreamer_filter_find (
            const char * name )
```

Find filter sub-plugin with the name.

Parameters

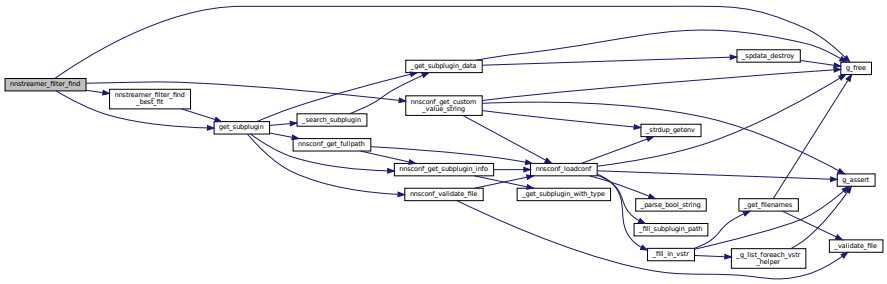
in	*name*	The name of filter sub-plugin.

Returns

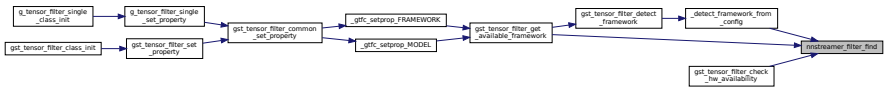
NULL if not found or the sub-plugin object has an error.

Definition at line 697 of file tensor_filter_common.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.106.3.62 nnstreamer_filter_find_best_fit()

```
static const GstTensorFilterFramework* nnstreamer_filter_find_best_fit (
            const char * names )  [static]
```

Find sub-plugin filter given the name list.

Parameters

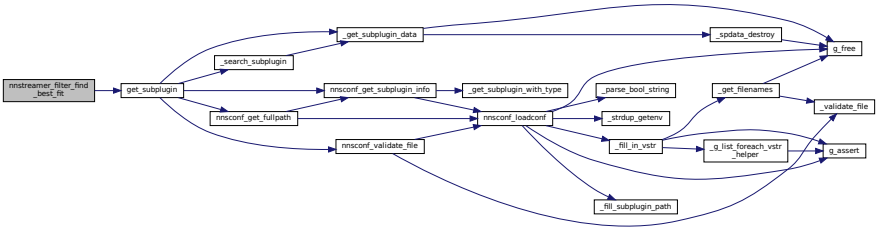
in	*names*	comma, whitespace separated list of the sub-plugin name

Returns

the best-fit sub-plugin object or NULL if not found.

Definition at line 664 of file tensor_filter_common.c.

Here is the call graph for this function:



Here is the caller graph for this function:

9.106.3.63 nnstreamer_filter_probe()

```
int nnstreamer_filter_probe (
            GstTensorFilterFramework * tfsp )
```

Filter's sub-plugin should call this function to register itself.

Parameters

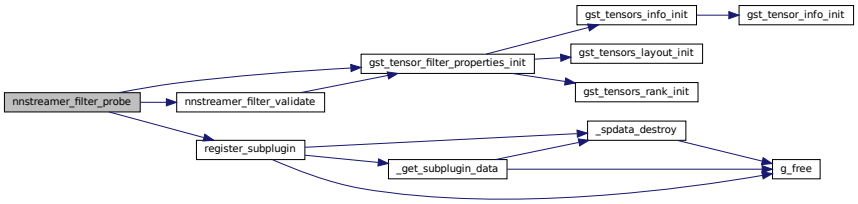
in	*tfsp*	Tensor-Filter Sub-Plugin to be registered.

Returns

TRUE if registered. FALSE is failed or duplicated.

Definition at line 611 of file tensor_filter_common.c.

Here is the call graph for this function:



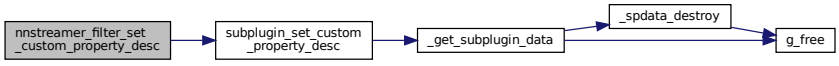
9.106.3.64 nnstreamer_filter_set_custom_property_desc()

```
void nnstreamer_filter_set_custom_property_desc (
            const char * name,
            const char * prop,
             ...  )
```

set custom property description for tensor filter sub-plugin

Definition at line 647 of file tensor_filter_common.c.

Here is the call graph for this function:



9.106.3.65 nnstreamer_filter_shared_model_get()

```
void* nnstreamer_filter_shared_model_get (
            void * instance,
            const char * key )
```

Get the shared model representation that is already shared and has the same key.

Parameters

in	*instance*	The instance that is sharing the model representation. It will be registered at the referred list.
in	*key*	The key to find the matched shared representation.

Returns

The model interpreter. NULL if it does not exist.

Definition at line 2993 of file tensor_filter_common.c.

9.106.3.66 nnstreamer_filter_shared_model_insert_and_get()

```
void* nnstreamer_filter_shared_model_insert_and_get (
            void * instance,
            char * key,
            void * interpreter )
```

Insert the new shared model representation and get the value.

Parameters

in	*instance*	The instance that is sharing the model representation. It will be registered at the referred list.
in	*key*	The key for shared model.
in	*interpreter*	The interpreter to be shared.

Returns

The model interpreter inserted. NULL if it is already inserted.

Internal error case. The interpreter already exists in shared table, do not insert and return null.

Definition at line 3026 of file tensor_filter_common.c.

9.106.3.67 nnstreamer_filter_shared_model_remove()

```
int nnstreamer_filter_shared_model_remove (
            void * instance,
            const char * key,
            void(*)(void *) free_callback )
```

Remove the instance registered at the referred list of shared model table. If referred list is empty, `free_callback` is executed.

Parameters

in	*instance*	The instance that should be removed from the referred list.
in	*key*	The key to find the shared model.
in	*free_callback*	The callback function to destroy the interpreter, which takes the interpreter as arg.

Returns

TRUE if the instance is removed. FALSE if failed to remove it.

Definition at line 3081 of file tensor_filter_common.c.

9.106.3.68 nnstreamer_filter_shared_model_replace()

```
void nnstreamer_filter_shared_model_replace (
            void * instance,
            const char * key,
            void * new_interpreter,
            void(*)(void *, void *) replace_callback,
            void(*)(void *) free_callback )
```

Helper to reload interpreter for instances that has shared key. `replace_callback` is called iterating instances in referred list.

Parameters

in	*instance*	The instance that is sharing the model representation.
in	*key*	The key to find the shared model.
in	*interpreter*	The new interpreter to replace.
in	*replace_callback*	The callback function to replace with new interpreter.
in	*free_callback*	The callback function to destroy the old interpreter.

Definition at line 3128 of file tensor_filter_common.c.

9.106.3.69 nnstreamer_filter_validate()

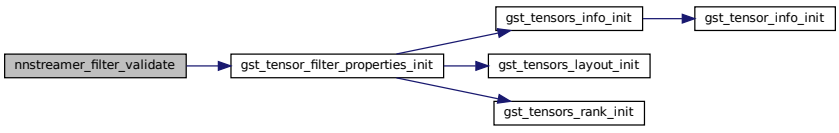
```
static gboolean nnstreamer_filter_validate (
            const GstTensorFilterFramework * tfsp )  [static]
```

Validate filter sub-plugin's data.

Mandatory callbacks are not defined

Definition at line 560 of file tensor_filter_common.c.

Here is the call graph for this function:



Here is the caller graph for this function:

9.106.3.70 parse_accl_hw_all()

```
static GList * parse_accl_hw_all (
            const gchar * accelerators,
            const gchar ** supported_accelerators )  [static]
```

parse user given string to extract list of accelerators based on given regex

Parameters

in	*accelerators*	user given input
in	*supported_accelerators*	list of supported accelerators

Returns

Corresponding list of accelerators maintaining given order

Note

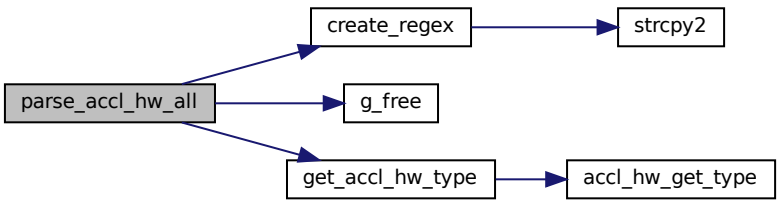
Returned list must be freed by the caller

Default to auto mode

Now match each provided element and get specific accelerator

Definition at line 2620 of file tensor_filter_common.c.

Here is the call graph for this function:



Here is the caller graph for this function:

9.106.3.71 parse_accl_hw_fill()

```
accl_hw parse_accl_hw_fill (
            parse_accl_args accl_args )
```

parse user given string to extract accelerator based on given regex filling in optional arguments

Note

The order of arguments for calling this function is:

- in_accl: user provided input accelerator string
- sup_accl: list of supported accelerator
- auto_accl: auto accelerator (optional)
- def_accl: default accelerator (optional)

remove unsupported accelerators from this list based on runtime system

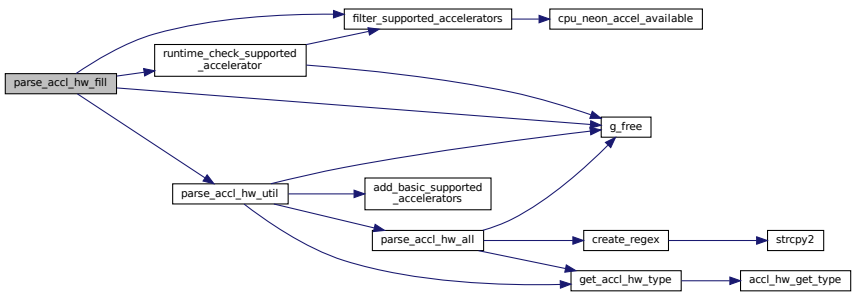
filtered supported accelerators can be empty

update default accelerator if it is not available at runtime

update auto accelerator if it is not available at runtime

Definition at line 2833 of file tensor_filter_common.c.

Here is the call graph for this function:



9.106.3.72 parse_accl_hw_util()

```
static accl_hw parse_accl_hw_util (
             const gchar * accelerators,
             const gchar ** supported_accelerators,
             const gchar * auto_accelerator,
             const gchar * default_accelerator )  [static]
```

parse user given string to extract accelerator based on given regex

Parameters

in	*accelerators*	user given input
in	*supported_accelerators*	list of supported accelerators
in	*auto_accelerator*	accelerator to use in auto case (when acceleration is enabled but specific accelerator is not provided or not matching)
in	*default_accelerator*	accelerator to use by default

Returns

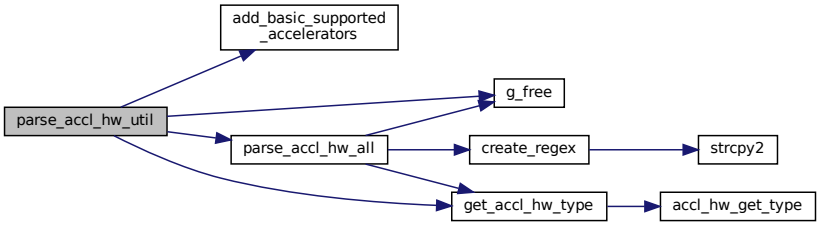
Corresponding accelerator. Returns ACCL_NONE if not found.

add auto and default accelerator into list of supported accelerators

This can be ACCL_NONE (no acceleration) or a specific accelerator

Definition at line 2763 of file tensor_filter_common.c.

Here is the call graph for this function:



Here is the caller graph for this function:

9.106.3.73 runtime_check_supported_accelerator()

```
static gint runtime_check_supported_accelerator (
            const gchar * accl )  [static]
```

Check if this accelerator can be used based on the runtime system.

Return values

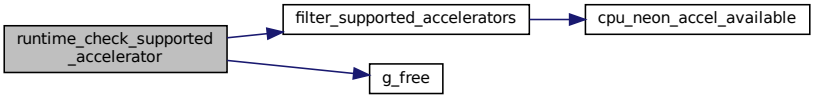
0	if filter can be used, -errno otherwise

Allocate the array

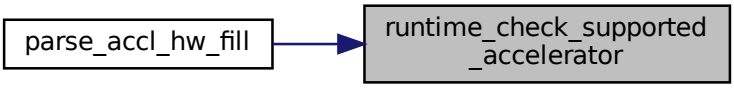
Fill the array

Definition at line 2804 of file tensor_filter_common.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.106.3.74 strcpy2()

```
static gchar* strcpy2 (
            gchar * dest,
            const gchar * src )  [static]
```

copy the string from src to destination

Parameters

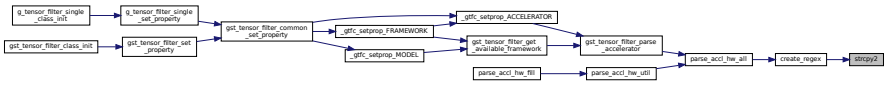
in	*dest*	destination string
in	*src*	source string

Returns

updated destination string

Definition at line 414 of file tensor_filter_common.c.

Here is the caller graph for this function:



9.106.3.75 verify_model_path()

```
static gboolean verify_model_path (
            const GstTensorFilterPrivate * priv )  [inline], [static]
```

Verify validity of path for given model file if verify_model_path is set.

Parameters

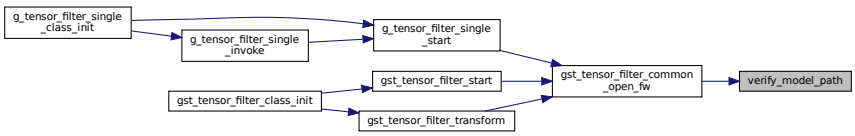
in	*priv*	Struct containing the common tensor-filter properties of the object

Returns

TRUE if there is no error

Definition at line 473 of file tensor_filter_common.c.

Here is the caller graph for this function:



9.106.4 Variable Documentation

9.106.4.1 regex_accl_elem_utils

```
const gchar* regex_accl_elem_utils[]  [static]
```

Initial value:

```
= {
  REGEX_ACCL_ELEM_START,
  REGEX_ACCL_ELEM_PREFIX,
  REGEX_ACCL_ELEM_SUFFIX,
  REGEX_ACCL_ELEM_DELIMITER,
  REGEX_ACCL_ELEM_END,
  NULL
}
```

Definition at line 75 of file tensor_filter_common.c.

9.106.4.2 regex_accl_utils

```
const gchar* regex_accl_utils[]  [static]
```

Initial value:

```
= {
  REGEX_ACCL_START,
  REGEX_ACCL_PREFIX,
  REGEX_ACCL_SUFFIX,
  REGEX_ACCL_DELIMITER,
  REGEX_ACCL_END,
  NULL
}
```

Definition at line 66 of file tensor_filter_common.c.

9.106.4.3 shared_model_table

```
GHashTable* shared_model_table = NULL  [static]
```

Definition at line 102 of file tensor_filter_common.c.

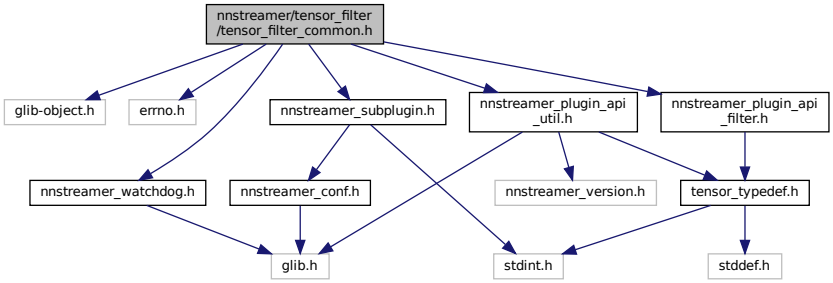
9.107 nnstreamer/tensor_filter/tensor_filter_common.h File Reference

Common functions for various tensor_filters.

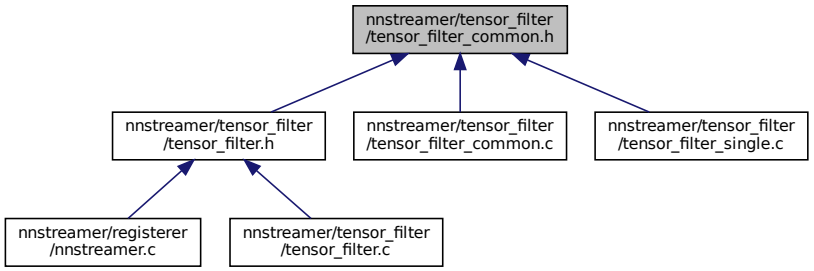
```
#include <glib-object.h>
#include <errno.h>
#include <nnstreamer_subplugin.h>
#include <nnstreamer_plugin_api_util.h>
#include <nnstreamer_plugin_api_filter.h>
```

```
#include "nnstreamer_watchdog.h"
```

Include dependency graph for tensor_filter_common.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct _GstTensorFilterStatistics

  Structure definition for tensor-filter statistics.
- struct _GstTensorFilterCombination

  Structure definition for tensor-filter in/out combination.
- struct GstTensorFilterSharedModelRepresentation

  Data Structure to store shared table.
- struct _GstTensorFilterPrivate

  Structure definition for common tensor-filter properties.

Macros

- #define DBG (!priv->silent)

  Macro for debug mode.
- #define GST_TF_FW_VN(fw, vn) (fw && checkGstTensorFilterFrameworkVersion (fw->version, vn))
- #define GST_TF_FW_V0(fw) GST_TF_FW_VN (fw, 0)
- #define GST_TF_FW_V1(fw) GST_TF_FW_VN (fw, 1)
- #define gst_tensor_filter_v0_call(priv, ret, funcname, ...)

  Invoke callbacks of nn framework. Guarantees calling open for the first call.
- #define gst_tensor_filter_v1_call(priv, ret, funcname, ...)
- #define GST_TF_FW_INVOKE_COMPAT(priv, ret, in, out)
- #define GST_TF_STAT_MAX_RECENT (10)

Typedefs

- typedef struct _GstTensorFilterStatistics GstTensorFilterStatistics

  Structure definition for tensor-filter statistics.
- typedef struct _GstTensorFilterCombination GstTensorFilterCombination

  Structure definition for tensor-filter in/out combination.
- typedef struct _GstTensorFilterPrivate GstTensorFilterPrivate

  Structure definition for common tensor-filter properties.

Enumerations

- enum {
  PROP_0, PROP_SILENT, PROP_FRAMEWORK, PROP_MODEL,
  PROP_INPUT, PROP_INPUTTYPE, PROP_INPUTNAME, PROP_INPUTLAYOUT,
  PROP_INPUTRANKS, PROP_OUTPUT, PROP_OUTPUTTYPE, PROP_OUTPUTNAME,
  PROP_OUTPUTLAYOUT, PROP_OUTPUTRANKS, PROP_CUSTOM, PROP_SUBPLUGINS,
  PROP_ACCELERATOR, PROP_IS_UPDATABLE, PROP_LATENCY, PROP_THROUGHPUT,
  PROP_INPUTCOMBINATION, PROP_OUTPUTCOMBINATION, PROP_SHARED_TENSOR_FILTER_KEY,
  PROP_LATENCY_REPORT,
  PROP_INVOKE_DYNAMIC, PROP_CONFIG, PROP_SUSPEND }

  GstTensorFilter properties.

Functions

- gchar * gst_tensorsinfo_compare_to_string (const GstTensorsInfo *info, const GstTensorsInfo *info2)

  Printout the comparison results of two tensors as a string.
- void gst_tensorsinfo_compare_print (const GstTensorsInfo *info1, const GstTensorsInfo *info2)

  Printout the comparison results of two tensors.
- gboolean gst_tensor_filter_allocate_in_invoke (GstTensorFilterPrivate *priv)

  check if the allocate_in_invoke is valid for the framework
- void gst_tensor_filter_install_properties (GObjectClass *gobject_class)

  Installs all the properties for tensor_filter.
- void gst_tensor_filter_common_init_property (GstTensorFilterPrivate *priv)

  Initialize the properties for tensor-filter.
- void gst_tensor_filter_common_free_property (GstTensorFilterPrivate *priv)

  Free the properties for tensor-filter.
- gboolean gst_tensor_filter_common_set_property (GstTensorFilterPrivate *priv, guint prop_id, const GValue *value, GParamSpec *pspec)

  Set the properties for tensor_filter.
- gboolean gst_tensor_filter_common_get_property (GstTensorFilterPrivate *priv, guint prop_id, GValue *value, GParamSpec *pspec)

  Get the properties for tensor_filter.
- gboolean gst_tensor_filter_common_get_combined_in_info (GstTensorFilterPrivate *priv, const GstTensorsInfo *in, GstTensorsInfo *combined)

  Configure input tensor info with combi option.
- gboolean gst_tensor_filter_common_get_combined_out_info (GstTensorFilterPrivate *priv, const GstTensorsInfo *in, const GstTensorsInfo *out, GstTensorsInfo *combined)

  Configure output tensor info with combi option.
- gboolean gst_tensor_filter_common_get_out_info (GstTensorFilterPrivate *priv, GstTensorsInfo *in, GstTensorsInfo *out)

  Get output tensor info from NN model with given input info.

- void gst_tensor_filter_load_tensor_info (GstTensorFilterPrivate *priv)

  Load tensor info from NN model. (both input and output tensor)
- void gst_tensor_filter_common_open_fw (GstTensorFilterPrivate *priv)

  Open NN framework.
- void gst_tensor_filter_common_unload_fw (GstTensorFilterPrivate *priv)

  Unload NN framework.
- void gst_tensor_filter_common_close_fw (GstTensorFilterPrivate *priv)

  Close NN framework.
- gchar * gst_tensor_filter_detect_framework (const gchar *const *model_files, const guint num_models, const gboolean load_conf)

  Get neural network framework name from given model file. This does not guarantee the framework is available on the target device.
- gboolean gst_tensor_filter_check_hw_availability (const gchar *name, const accl_hw hw, const char *custom)

  Check if the given hw is supported by the framework.
- void gst_tensor_filter_destroy_notify_util (GstTensorFilterPrivate *priv, void *data)

  Free the data allocated for tensor filter output.

9.107.1 Detailed Description

Common functions for various tensor_filters.

Copyright (C) 2019 Parichay kapoor pk.kapoor@samsung.com

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Library General Public License as published by the Free Software Foundation; version 2.1 of the License.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Library General Public License for more details.

Date

28 Aug 2019

See also

http://github.com/nnstreamer/nnstreamer

Author

Parichay Kapoor pk.kapoor@samsung.com

MyungJoo Ham myungjoo.ham@samsung.com

Bug No known bugs except for NYI items

9.107.2 Macro Definition Documentation

9.107.2.1 DBG

```
#define DBG (!priv->silent)
```

Macro for debug mode.

Definition at line 41 of file tensor_filter_common.h.

9.107.2.2 gst_tensor_filter_v0_call

```
#define gst_tensor_filter_v0_call(
              priv,
              ret,
              funcname,
              ...  )
```

Value:

```
      do { \
      gst_tensor_filter_common_open_fw (priv); \
      ret = -1; \
      if ((priv)->prop.fw_opened && (priv)->fw && (priv)->fw->funcname) { \
        ret = (priv)->fw->funcname (&(priv)->prop, &(priv)->privateData, __VA_ARGS__); \
      } \
    } while (0)
```

Invoke callbacks of nn framework. Guarantees calling open for the first call.

Definition at line 53 of file tensor_filter_common.h.

9.107.2.3 gst_tensor_filter_v1_call

```
#define gst_tensor_filter_v1_call(
              priv,
              ret,
              funcname,
              ...  )
```

Value:

```
      do { \
      gst_tensor_filter_common_open_fw (priv); \
      ret = -1; \
      if ((priv)->prop.fw_opened && (priv)->fw && (priv)->fw->funcname) { \
        ret = (priv)->fw->funcname ((priv)->fw, &(priv)->prop, (priv)->privateData, __VA_ARGS__); \
      } \
    } while (0)
```

Definition at line 61 of file tensor_filter_common.h.

9.107.2.4 GST_TF_FW_INVOKE_COMPAT

```
#define GST_TF_FW_INVOKE_COMPAT(
              priv,
              ret,
              in,
              out )
```

Value:

```
      do { \
      ret = -1; \
      if (GST_TF_FW_V0 ((priv)->fw)) { \
        ret = (priv)->fw->invoke_NN (&(priv)->prop, &(priv)->privateData, (in), (out)); \
      } else if (GST_TF_FW_V1 ((priv)->fw)) { \
        ret = (priv)->fw->invoke ((priv)->fw, &(priv)->prop, (priv)->privateData, (in), (out)); \
      } \
    } while (0)
```

Definition at line 69 of file tensor_filter_common.h.

9.107.2.5 GST_TF_FW_V0

```
#define GST_TF_FW_V0(
              fw ) GST_TF_FW_VN (fw, 0)
```

Definition at line 47 of file tensor_filter_common.h.

9.107.2.6 GST_TF_FW_V1

```
#define GST_TF_FW_V1(
              fw ) GST_TF_FW_VN (fw, 1)
```

Definition at line 48 of file tensor_filter_common.h.

9.107.2.7 GST_TF_FW_VN

```
#define GST_TF_FW_VN(
              fw,
              vn ) (fw && checkGstTensorFilterFrameworkVersion (fw->version, vn))
```

Check tensor_filter framework version

Definition at line 45 of file tensor_filter_common.h.

9.107.2.8 GST_TF_STAT_MAX_RECENT

```
#define GST_TF_STAT_MAX_RECENT (10)
```

Definition at line 78 of file tensor_filter_common.h.

9.107.3 Typedef Documentation

9.107.3.1 GstTensorFilterCombination

```
typedef struct _GstTensorFilterCombination GstTensorFilterCombination
```

Structure definition for tensor-filter in/out combination.

9.107.3.2 GstTensorFilterPrivate

```
typedef struct _GstTensorFilterPrivate GstTensorFilterPrivate
```

Structure definition for common tensor-filter properties.

9.107.3.3 GstTensorFilterStatistics

```
typedef struct _GstTensorFilterStatistics GstTensorFilterStatistics
```

Structure definition for tensor-filter statistics.

9.107.4 Enumeration Type Documentation

Enumerator

9.107.4.1 anonymous enum

```
anonymous enum
```

GstTensorFilter properties.

Enumerator

PROP_0	
PROP_SILENT	
PROP_FRAMEWORK	
PROP_MODEL	
PROP_INPUT	
PROP_INPUTTYPE	
PROP_INPUTNAME	
PROP_INPUTLAYOUT	
PROP_INPUTRANKS	
PROP_OUTPUT	
PROP_OUTPUTTYPE	
PROP_OUTPUTNAME	
PROP_OUTPUTLAYOUT	
PROP_OUTPUTRANKS	
PROP_CUSTOM	
PROP_SUBPLUGINS	
PROP_ACCELERATOR	
PROP_IS_UPDATABLE	
PROP_LATENCY	
PROP_THROUGHPUT	
PROP_INPUTCOMBINATION	
PROP_OUTPUTCOMBINATION	
PROP_SHARED_TENSOR_FILTER_KEY	
PROP_LATENCY_REPORT	
PROP_INVOKE_DYNAMIC	
PROP_CONFIG	
PROP_SUSPEND	

Definition at line 83 of file tensor_filter_common.h.

9.107.5 Function Documentation

9.107.5.1 gst_tensor_filter_allocate_in_invoke()

```
gboolean gst_tensor_filter_allocate_in_invoke (
            GstTensorFilterPrivate * priv )
```

check if the allocate_in_invoke is valid for the framework

Parameters

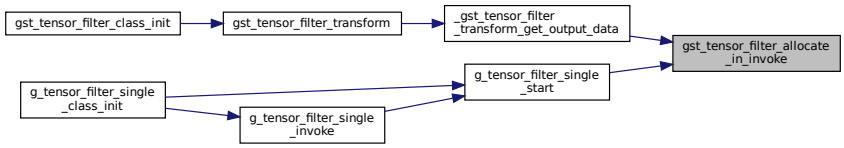
in	*priv*	Struct containing the properties of the object

Returns

TRUE if valid, FALSE on error

Definition at line 757 of file tensor_filter_common.c.

Here is the caller graph for this function:



9.107.5.2 gst_tensor_filter_check_hw_availability()

```
gboolean gst_tensor_filter_check_hw_availability (
            const gchar * name,
            const accl_hw hw,
            const char * custom )
```

Check if the given hw is supported by the framework.

Parameters

in	*name*	The name of filter sub-plugin.
in	*hw*	Backend accelerator hardware.
in	*custom*	User-defined string to handle detailed hardware option.

Returns

TRUE if given hw is available.

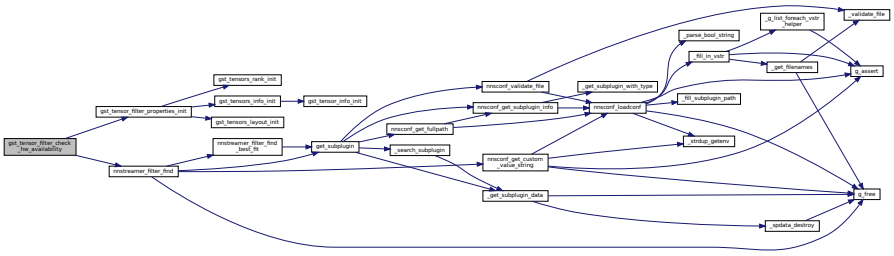
Note

This function is included in nnstreamer internal header for native APIs. When changing the declaration, you should update the internal header (nnstreamer_internal.h).

Only check for specific HW, DEFAULT/AUTO are always supported

Definition at line 2923 of file tensor_filter_common.c.

Here is the call graph for this function:



9.107.5.3 gst_tensor_filter_common_close_fw()

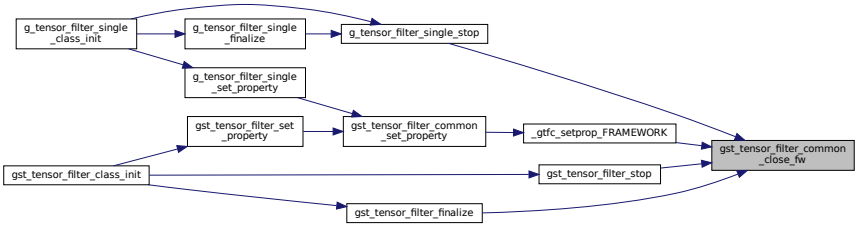
```
void gst_tensor_filter_common_close_fw (
            GstTensorFilterPrivate * priv )
```

Close NN framework.

Definition at line 2560 of file tensor_filter_common.c.

Here is the call graph for this function:

Here is the caller graph for this function:



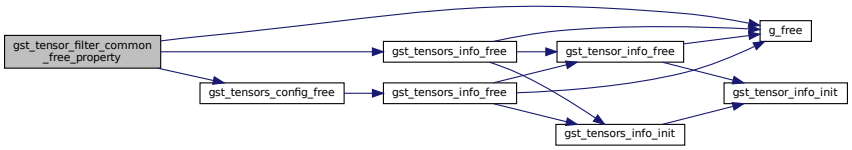
9.107.5.4 gst_tensor_filter_common_free_property()

```
void gst_tensor_filter_common_free_property (
            GstTensorFilterPrivate * priv )
```

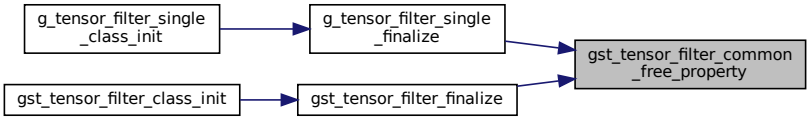
Free the properties for tensor-filter.

Definition at line 1065 of file tensor_filter_common.c.

Here is the call graph for this function:



Here is the caller graph for this function:



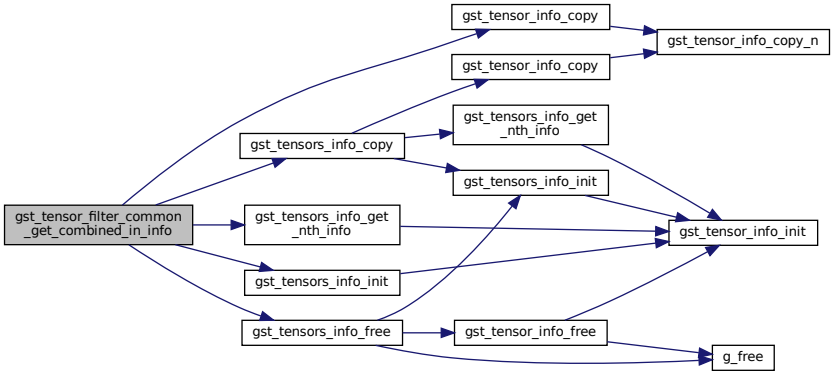
9.107.5.5 gst_tensor_filter_common_get_combined_in_info()

```
gboolean gst_tensor_filter_common_get_combined_in_info (
            GstTensorFilterPrivate * priv,
            const GstTensorsInfo * in,
            GstTensorsInfo * combined )
```

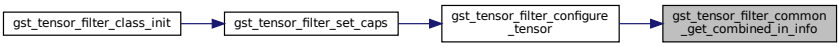
Configure input tensor info with combi option.

Definition at line 2270 of file tensor_filter_common.c.

Here is the call graph for this function:



Here is the caller graph for this function:



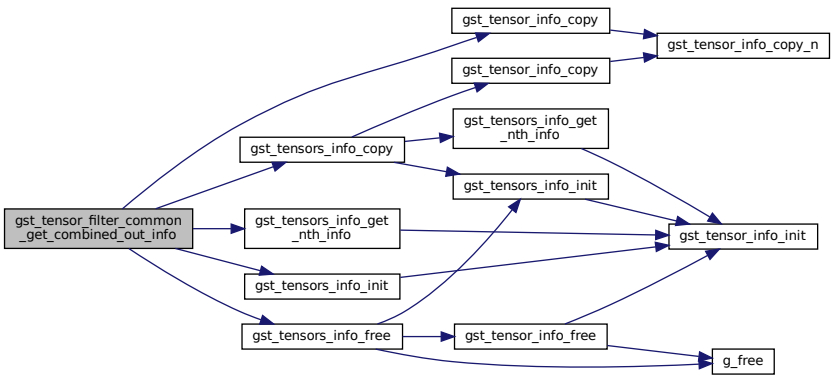
9.107.5.6 gst_tensor_filter_common_get_combined_out_info()

```
gboolean gst_tensor_filter_common_get_combined_out_info (
            GstTensorFilterPrivate * priv,
            const GstTensorsInfo * in,
            const GstTensorsInfo * out,
            GstTensorsInfo * combined )
```

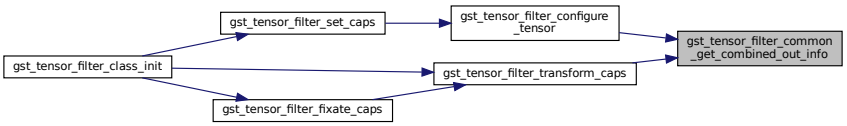
Configure output tensor info with combi option.

Definition at line 2315 of file tensor_filter_common.c.

Here is the call graph for this function:



Here is the caller graph for this function:



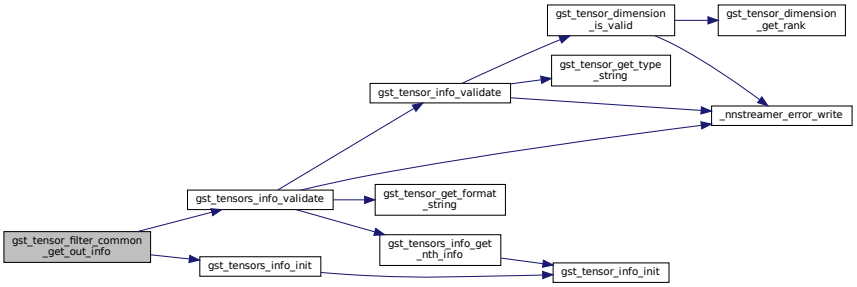
9.107.5.7 gst_tensor_filter_common_get_out_info()

```
gboolean gst_tensor_filter_common_get_out_info (
            GstTensorFilterPrivate * priv,
            GstTensorsInfo * in,
            GstTensorsInfo * out )
```

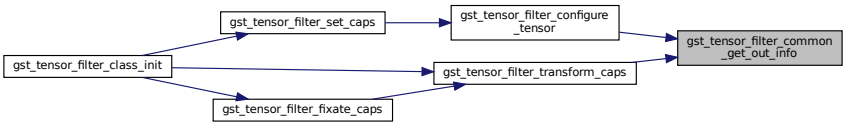
Get output tensor info from NN model with given input info.

Definition at line 2374 of file tensor_filter_common.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.107.5.8 gst_tensor_filter_common_get_property()

```
gboolean gst_tensor_filter_common_get_property (
            GstTensorFilterPrivate * priv,
            guint prop_id,
            GValue * value,
            GParamSpec * pspec )
```

Get the properties for tensor_filter.

Parameters

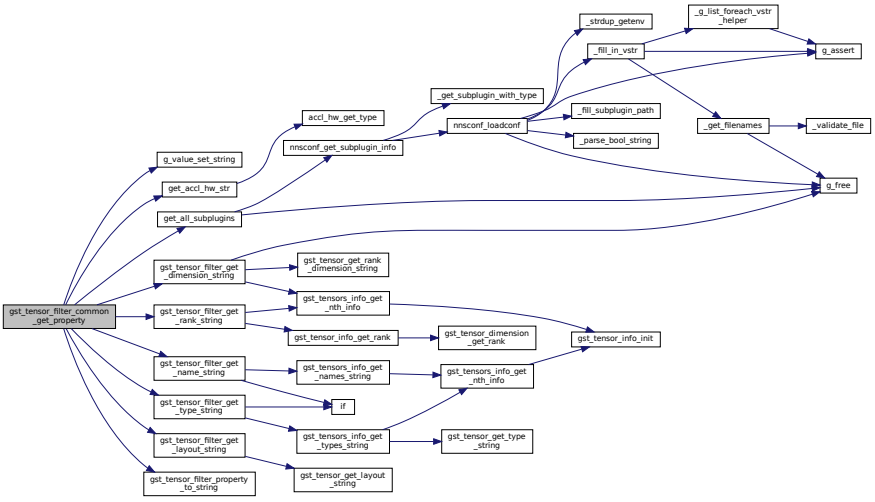
in	*priv*	Struct containing the properties of the object
in	*prop_id*	Id for the property
in	*value*	Container to return the asked property
in	*pspec*	Metadata to specify the parameter

Returns

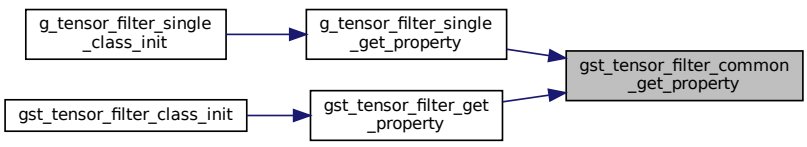
TRUE if prop_id is value, else FALSE

Definition at line 2102 of file tensor_filter_common.c.

Here is the call graph for this function:



Here is the caller graph for this function:



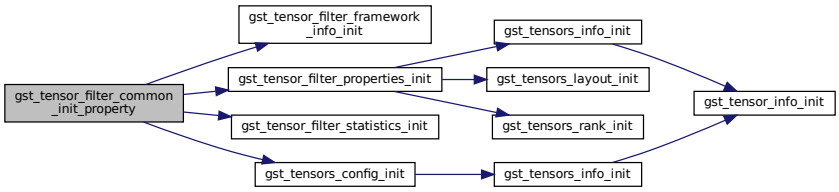
9.107.5.9 gst_tensor_filter_common_init_property()

```
void gst_tensor_filter_common_init_property (
            GstTensorFilterPrivate * priv )
```

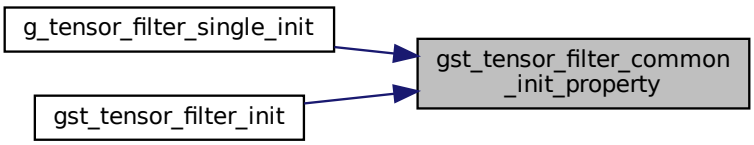
Initialize the properties for tensor-filter.

Definition at line 1041 of file tensor_filter_common.c.

Here is the call graph for this function:



Here is the caller graph for this function:



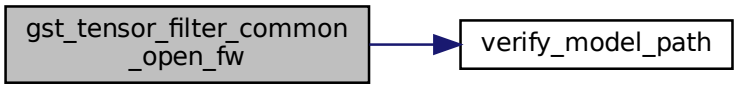
9.107.5.10 gst_tensor_filter_common_open_fw()

```
void gst_tensor_filter_common_open_fw (
            GstTensorFilterPrivate * priv )
```

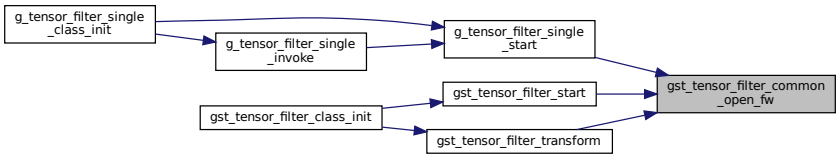
Open NN framework.

Definition at line 2491 of file tensor_filter_common.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.107.5.11 gst_tensor_filter_common_set_property()

```
gboolean gst_tensor_filter_common_set_property (
            GstTensorFilterPrivate * priv,
            guint prop_id,
            const GValue * value,
            GParamSpec * pspec )
```

Set the properties for tensor_filter.

Parameters

in	*priv*	Struct containing the properties of the object
in	*prop↩ _id*	Id for the property
in	*value*	Container to return the asked property
in	*pspec*	Metadata to specify the parameter

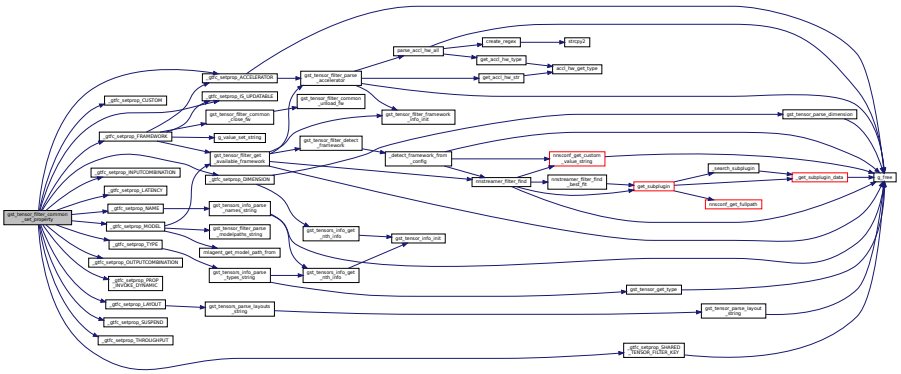
Returns

TRUE if prop_id is value, else FALSE

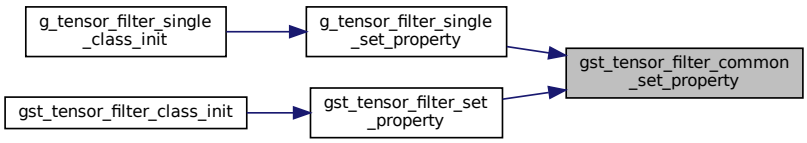
Although no one return !0 at this point, let's enable error handling.

Definition at line 1967 of file tensor_filter_common.c.

Here is the call graph for this function:



Here is the caller graph for this function:



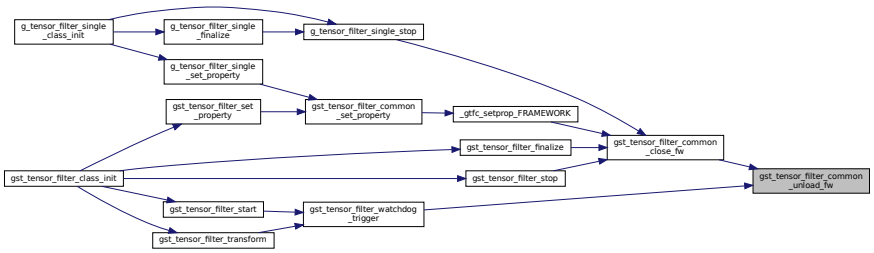
9.107.5.12 gst_tensor_filter_common_unload_fw()

```
void gst_tensor_filter_common_unload_fw (
            GstTensorFilterPrivate * priv )
```

Unload NN framework.

Definition at line 2545 of file tensor_filter_common.c.

Here is the caller graph for this function:



9.107.5.13 gst_tensor_filter_destroy_notify_util()

```
void gst_tensor_filter_destroy_notify_util (
            GstTensorFilterPrivate * priv,
            void * data )
```

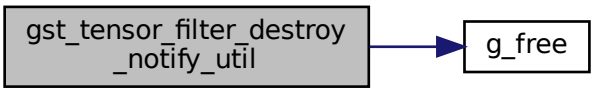
Free the data allocated for tensor filter output.

Parameters

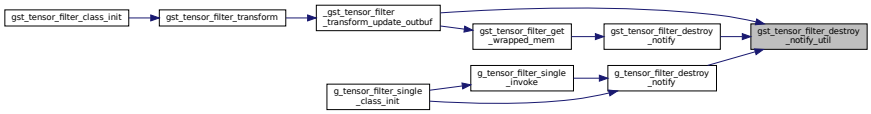
in	*priv*	Struct containing the properties of the object
in	*data*	Data to be freed

Definition at line 786 of file tensor_filter_common.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.107.5.14 gst_tensor_filter_detect_framework()

```
gchar* gst_tensor_filter_detect_framework (
            const gchar *const * model_files,
            const guint num_models,
            const gboolean load_conf )
```

Get neural network framework name from given model file. This does not guarantee the framework is available on the target device.

Parameters

in	*model_files*	the prediction model paths
in	*num_models*	the number of model files
in	*load_conf*	flag to load configuration for the priority of framework

Returns

Possible framework name (NULL if it fails to detect automatically). Caller should free returned value using g_free().

Parameters

in	*model_files*	the prediction model paths
in	*num_models*	the number of model files
in	*load_conf*	flag to load configuration for the priority of framework

Returns

Possible framework name (NULL if it fails to detect automatically). Caller should free returned value using g_free().

Note

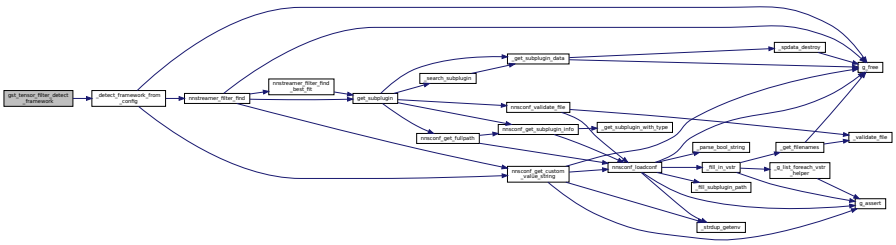
This function is included in nnstreamer internal header for native APIs. When changing the declaration, you should update the internal header (nnstreamer_internal.h).

Note

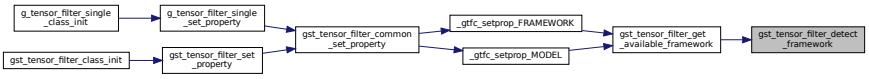
When adding new file extension for auto fw detection, you should check the condition to validate model file - ml_validate_model_file() in ML-API.

Definition at line 1232 of file tensor_filter_common.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.107.5.15 gst_tensor_filter_install_properties()

```
void gst_tensor_filter_install_properties (
            GObjectClass * gobject_class )
```

Installs all the properties for tensor_filter.

Parameters

in	*gobject_class*	Glib object class whose properties will be set

min

max

default: off

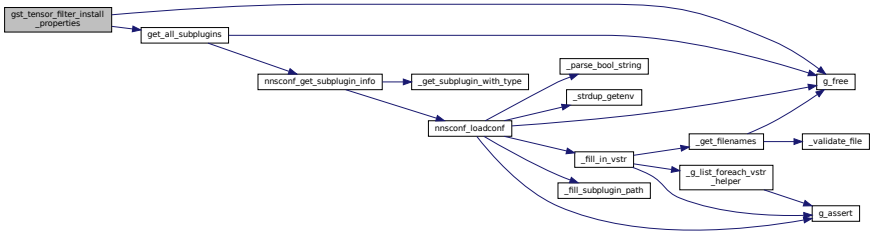
min

max

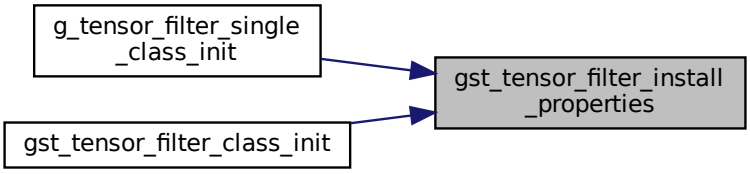
default: off

Definition at line 888 of file tensor_filter_common.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.107.5.16 gst_tensor_filter_load_tensor_info()

```
void gst_tensor_filter_load_tensor_info (
            GstTensorFilterPrivate * priv )
```

Load tensor info from NN model. (both input and output tensor)

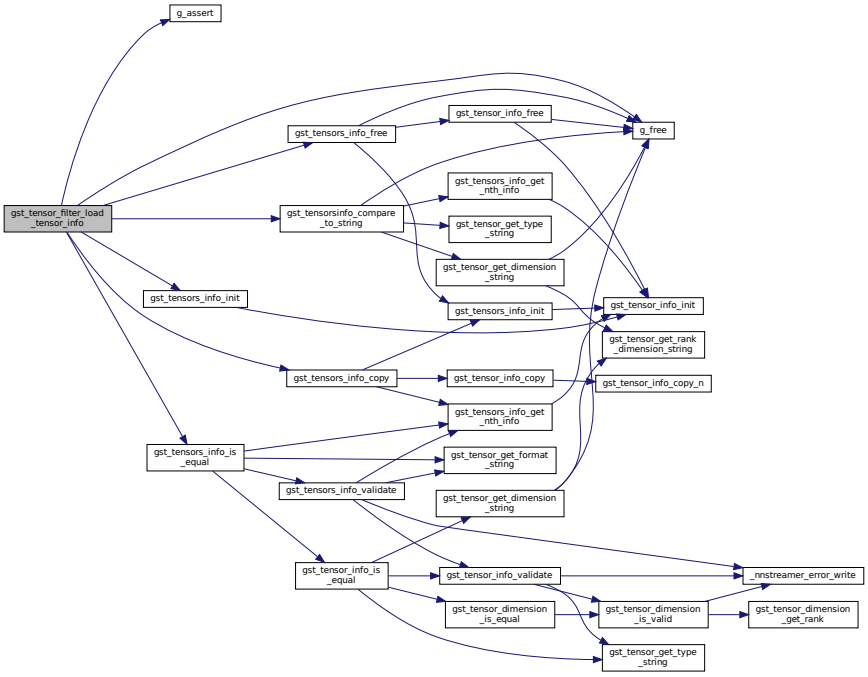
if set-property called and already has info, verify it!

In case of dynamic invoke, output tensors info is determined after invoke.

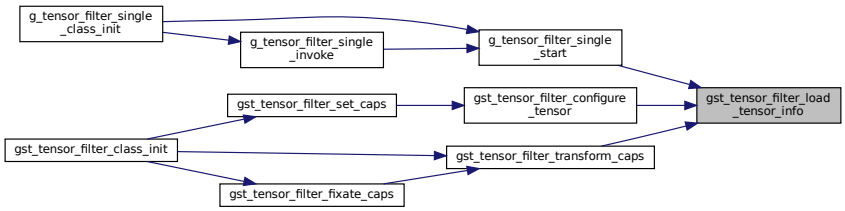
if set-property called and already has info, verify it!

Definition at line 2409 of file tensor_filter_common.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.107.5.17 gst_tensorsinfo_compare_print()

```
void gst_tensorsinfo_compare_print (
            const GstTensorsInfo * info1,
            const GstTensorsInfo * info2 )
```

Printout the comparison results of two tensors.

Parameters

in	*info1*	The tensors to be shown on the left hand side
in	*info2*	The tensors to be shown on the right hand side

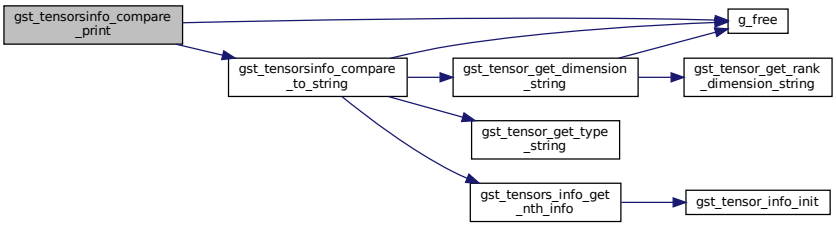
Todo If this is going to be used by other elements, move this to nnstreamer/tensor_common.

Parameters

in	*info1*	The tensors to be shown on the left hand side
in	*info2*	The tensors to be shown on the right hand side

Definition at line 874 of file tensor_filter_common.c.

Here is the call graph for this function:



9.107.5.18 gst_tensorsinfo_compare_to_string()

```
gchar* gst_tensorsinfo_compare_to_string (
            const GstTensorsInfo * info1,
            const GstTensorsInfo * info2 )
```

Printout the comparison results of two tensors as a string.

Parameters

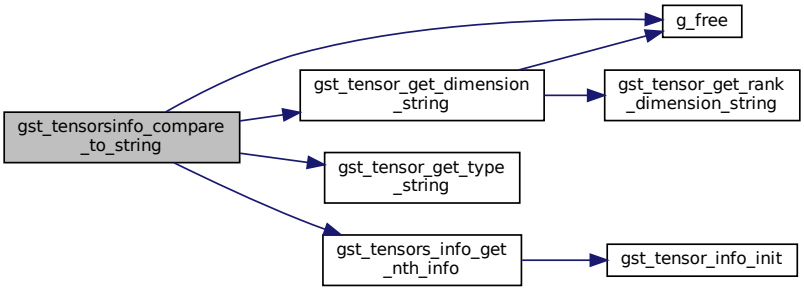
in	*info1*	The tensors to be shown on the left hand side
in	*info2*	The tensors to be shown on the right hand side

Returns

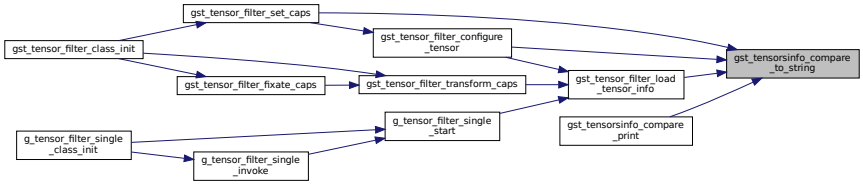
The printout string allocated. Caller should free the value.

Definition at line 811 of file tensor_filter_common.c.

Here is the call graph for this function:



Here is the caller graph for this function:



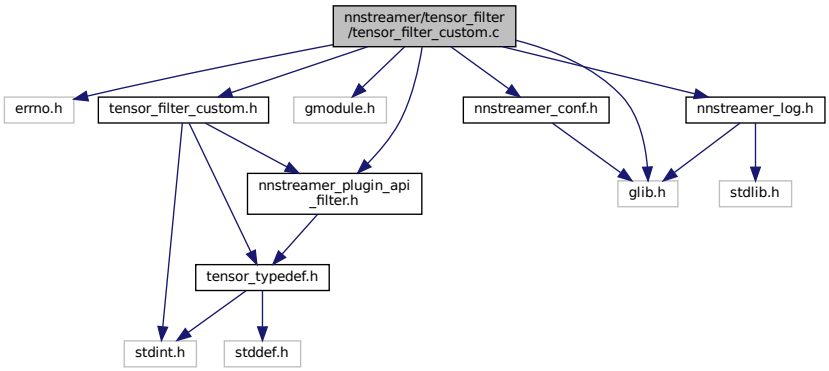
9.108 nnstreamer/tensor_filter/tensor_filter_custom.c File Reference

Custom tensor post-processing interface for NNStreamer suite between NN developer-plugins and NNstreamer.

```
#include <errno.h>
#include <glib.h>
#include <gmodule.h>
#include "tensor_filter_custom.h"
#include "nnstreamer_plugin_api_filter.h"
#include "nnstreamer_conf.h"
```

```
#include <nnstreamer_log.h>
```

Include dependency graph for tensor_filter_custom.c:



Classes

- struct _internal_data

  internal_data

Typedefs

- typedef struct _internal_data internal_data

Functions

- void init_filter_custom (void)

  Initialize this object for tensor_filter subplugin runtime register.
- static int custom_loadlib (const GstTensorFilterProperties *prop, void **private_data)

  Load the custom library. Will skip loading if it's already loaded.
- static int custom_open (const GstTensorFilterProperties *prop, void **private_data)

  The open callback for GstTensorFilterFramework. Called before anything else.
- static int custom_invoke (const GstTensorFilterProperties *prop, void **private_data, const GstTensorMemory *input, GstTensorMemory *output)

  The mandatory callback for GstTensorFilterFramework.
- static int custom_getInputDim (const GstTensorFilterProperties *prop, void **private_data, GstTensorsInfo *info)

  The optional callback for GstTensorFilterFramework.
- static int custom_getOutputDim (const GstTensorFilterProperties *prop, void **private_data, GstTensorsInfo *info)

  The optional callback for GstTensorFilterFramework.
- static int custom_setInputDim (const GstTensorFilterProperties *prop, void **private_data, const GstTensorsInfo *in_info, GstTensorsInfo *out_info)

  The set-input-dim callback for GstTensorFilterFramework.
- static void custom_close (const GstTensorFilterProperties *prop, void **private_data)

  Free privateData and move on.
- static void custom_destroyNotify (void **private_data, void *data)

The optional callback for GstTensorFilterFramework.

- static int custom_allocateInInvoke (void **private_data)

  The optional callback for GstTensorFilterFramework.

- static int custom_checkAvailability (accl_hw hw)

  Check support of the backend.

- void fini_filter_custom (void)

  Destruct the subplugin.

Variables

- static gchar filter_subplugin_custom [ ] = "custom"
- static GstTensorFilterFramework NNS_support_custom

9.108.1 Detailed Description

Custom tensor post-processing interface for NNStreamer suite between NN developer-plugins and NNstreamer.

GStreamer Tensor_Filter Module Copyright (C) 2018 MyungJoo Ham myungjoo.ham@samsung.com

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Library General Public License as published by the Free Software Foundation; version 2.1 of the License.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Library General Public License for more details.

Date

01 Jun 2018

See also

http://github.com/nnstreamer/nnstreamer

Author

MyungJoo Ham myungjoo.ham@samsung.com

Bug No known bugs except for NYI items

This is the per-NN-framework plugin (custom) for tensor_filter. Fill in "GstTensorFilterFramework" for tensor_filter.h/c

9.108.2 Typedef Documentation

9.108.2.1 internal_data

```
typedef struct _internal_data internal_data
```

Definition at line 57 of file tensor_filter_custom.c.

9.108.3 Function Documentation

9.108.3.1 custom_allocateInInvoke()

```
static int custom_allocateInInvoke (
            void ** private_data )  [static]
```

The optional callback for GstTensorFilterFramework.

Definition at line 283 of file tensor_filter_custom.c.

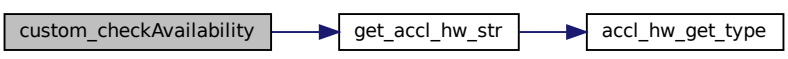
9.108.3.2 custom_checkAvailability()

```
static int custom_checkAvailability (
            accl_hw hw )  [static]
```

Check support of the backend.

Definition at line 298 of file tensor_filter_custom.c.

Here is the call graph for this function:



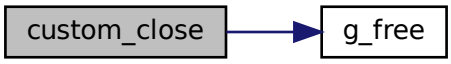
9.108.3.3 custom_close()

```
static void custom_close (
             const GstTensorFilterProperties * prop,
             void ** private_data )  [static]
```

Free privateData and move on.

Definition at line 252 of file tensor_filter_custom.c.

Here is the call graph for this function:



9.108.3.4 custom_destroyNotify()

```
static void custom_destroyNotify (
             void ** private_data,
             void * data )  [static]
```

The optional callback for GstTensorFilterFramework.

Definition at line 267 of file tensor_filter_custom.c.

Here is the call graph for this function:

9.108.3.5 custom_getInputDim()

```
static int custom_getInputDim (
             const GstTensorFilterProperties * prop,
             void ** private_data,
             GstTensorsInfo * info )  [static]
```

The optional callback for GstTensorFilterFramework.

Definition at line 193 of file tensor_filter_custom.c.

9.108.3.6 custom_getOutputDim()

```
static int custom_getOutputDim (
             const GstTensorFilterProperties * prop,
             void ** private_data,
             GstTensorsInfo * info )  [static]
```

The optional callback for GstTensorFilterFramework.

Definition at line 212 of file tensor_filter_custom.c.

9.108.3.7 custom_invoke()

```
static int custom_invoke (
             const GstTensorFilterProperties * prop,
             void ** private_data,
             const GstTensorMemory * input,
             GstTensorMemory * output )  [static]
```

The mandatory callback for GstTensorFilterFramework.

Parameters

	prop	The properties of parent object
in	*input*	The array of input tensors
out	*output*	The array of output tensors

Returns

0 if OK. non-zero if error.

Definition at line 167 of file tensor_filter_custom.c.

9.108.3.8 custom_loadlib()

```
static int custom_loadlib (
            const GstTensorFilterProperties * prop,
            void ** private_data )  [static]
```

Load the custom library. Will skip loading if it's already loaded.

Returns

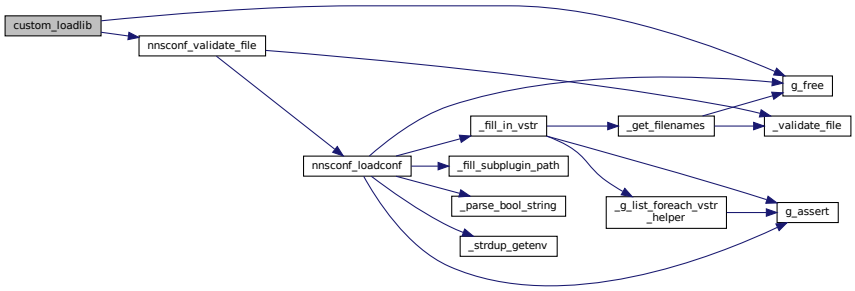
0 if successfully loaded. 1 if skipped (already loaded). -1 if error

Todo : Check the integrity of filter->data and filter->model_file, nnfw

Todo Double check if this check is really required and safe

Definition at line 64 of file tensor_filter_custom.c.

Here is the call graph for this function:



Here is the caller graph for this function:

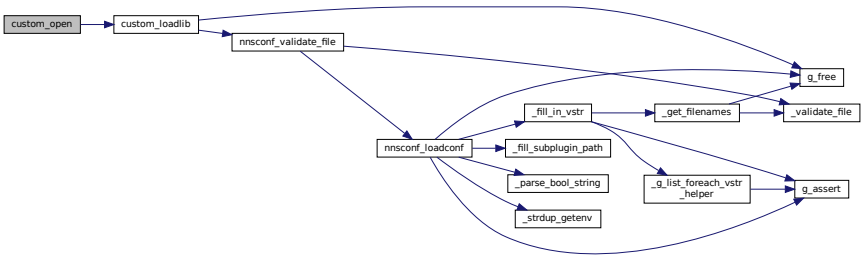
9.108.3.9 custom_open()

```
static int custom_open (
            const GstTensorFilterProperties * prop,
            void ** private_data )  [static]
```

The open callback for GstTensorFilterFramework. Called before anything else.

Definition at line 138 of file tensor_filter_custom.c.

Here is the call graph for this function:



9.108.3.10 custom_setInputDim()

```
static int custom_setInputDim (
            const GstTensorFilterProperties * prop,
            void ** private_data,
            const GstTensorsInfo * in_info,
            GstTensorsInfo * out_info )  [static]
```

The set-input-dim callback for GstTensorFilterFramework.

Definition at line 231 of file tensor_filter_custom.c.

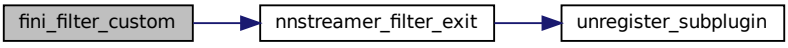
9.108.3.11 fini_filter_custom()

```
void fini_filter_custom (
            void  )
```

Destruct the subplugin.

Definition at line 335 of file tensor_filter_custom.c.

Here is the call graph for this function:



9.108.3.12 init_filter_custom()

```
void init_filter_custom (
             void  )
```

Initialize this object for tensor_filter subplugin runtime register.

Definition at line 38 of file tensor_filter_custom.c.

9.108.4 Variable Documentation

9.108.4.1 filter_subplugin_custom

```
gchar filter_subplugin_custom[] = "custom"  [static]
```

Definition at line 306 of file tensor_filter_custom.c.

9.108.4.2 NNS_support_custom

```
GstTensorFilterFramework NNS_support_custom  [static]
```

Initial value:

```
= {
  .version = GST_TENSOR_FILTER_FRAMEWORK_V0,
  .name = filter_subplugin_custom,
  .allow_in_place = FALSE,
  .allocate_in_invoke = TRUE,
  .run_without_model = FALSE,
  .invoke_NN = custom_invoke,
  .handleEvent = NULL,
  .getInputDimension = custom_getInputDim,
  .getOutputDimension = custom_getOutputDim,
  .setInputDimension = custom_setInputDim,
  .open = custom_open,
  .close = custom_close,
  .destroyNotify = custom_destroyNotify,
  .allocateInInvoke = custom_allocateInInvoke,
  .checkAvailability = custom_checkAvailability,
}
```

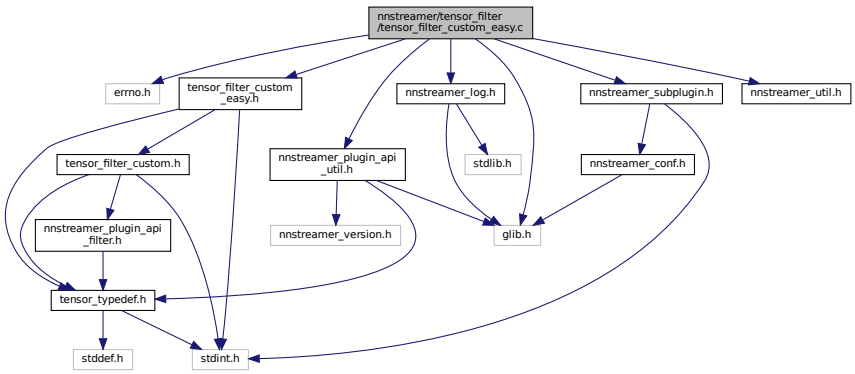
Definition at line 308 of file tensor_filter_custom.c.

9.109 nnstreamer/tensor_filter/tensor_filter_custom_easy.c File Reference

Custom tensor processing interface for simple functions.

```
#include <errno.h>
#include <glib.h>
#include <tensor_filter_custom_easy.h>
#include <nnstreamer_log.h>
#include <nnstreamer_plugin_api_util.h>
#include <nnstreamer_subplugin.h>
#include <nnstreamer_util.h>
```

Include dependency graph for tensor_filter_custom_easy.c:



Classes

- struct runtime_data

  The easy-filter user's data.

Functions

- void init_filter_custom_easy (void)

  internal_data
- static void custom_free_internal_data (internal_data *data)

  Internal function to release internal data.
- int NNS_custom_easy_register (const char *modelname, NNS_custom_invoke func, void *data, const GstTensorsInfo *in_info, const GstTensorsInfo *out_info)

  Register the custom-easy tensor function. More info in .h.
- int NNS_custom_easy_dynamic_register (const char *modelname, NNS_custom_invoke_dynamic func, void *data, const GstTensorsInfo *in_info)

  Register the custom-easy tensor function. More info in .h.
- int NNS_custom_easy_unregister (const char *modelname)

  Unregister the custom-easy tensor function.
- static int custom_open (const GstTensorFilterProperties *prop, void **private_data)

  Callback required by tensor_filter subplugin.
- static void custom_close (const GstTensorFilterProperties *prop, void **private_data)

Callback required by tensor_filter subplugin.

- static int custom_invoke (const GstTensorFilterFramework *self, GstTensorFilterProperties *prop, void *private_data, const GstTensorMemory *input, GstTensorMemory *output)

  Callback required by tensor_filter subplugin.
- static int custom_getFrameworkInfo (const GstTensorFilterFramework *self, const GstTensorFilterProperties *prop, void *private_data, GstTensorFilterFrameworkInfo *fw_info)

  V1 tensor-filter wrapper callback function, "getFrameworkInfo".
- static int custom_getModelInfo (const GstTensorFilterFramework *self, const GstTensorFilterProperties *prop, void *private_data, model_info_ops ops, GstTensorsInfo *in_info, GstTensorsInfo *out_info)

  C V1 tensor-filter wrapper callback function, "getModelInfo".
- static int custom_eventHandler (const GstTensorFilterFramework *self, const GstTensorFilterProperties *prop, void *private_data, event_ops ops, GstTensorFilterFrameworkEventData *data)

  C V1 tensor-filter wrapper callback function, "eventHandler".
- void fini_filter_custom_easy (void)

  Destruct the subplugin.

Variables

- internal_data
- static GstTensorFilterFramework NNS_support_custom_easy

9.109.1 Detailed Description

Custom tensor processing interface for simple functions.

GStreamer Tensor_Filter, Customized Module, Easy Mode Copyright (C) 2019 MyungJoo Ham myungjoo.ham@samsung.com

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Library General Public License as published by the Free Software Foundation; version 2.1 of the License.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Library General Public License for more details.

Date

24 Oct 2019

See also

http://github.com/nnstreamer/nnstreamer

Author

MyungJoo Ham myungjoo.ham@samsung.com

Bug No known bugs except for NYI items

9.109.2 Function Documentation

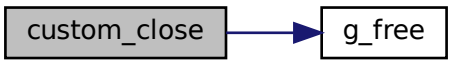
9.109.2.1 custom_close()

```
static void custom_close (
             const GstTensorFilterProperties * prop,
             void ** private_data )  [static]
```

Callback required by tensor_filter subplugin.

Definition at line 230 of file tensor_filter_custom_easy.c.

Here is the call graph for this function:



9.109.2.2 custom_eventHandler()

```
static int custom_eventHandler (
             const GstTensorFilterFramework * self,
             const GstTensorFilterProperties * prop,
             void * private_data,
             event_ops ops,
             GstTensorFilterFrameworkEventData * data )  [static]
```

C V1 tensor-filter wrapper callback function, "eventHandler".

Definition at line 319 of file tensor_filter_custom_easy.c.

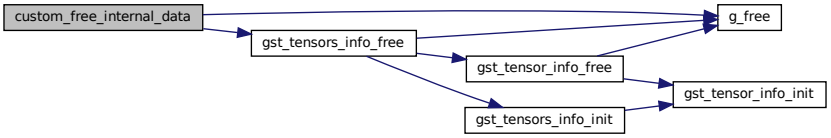
9.109.2.3 custom_free_internal_data()

```
static void custom_free_internal_data (
            internal_data * data )  [static]
```

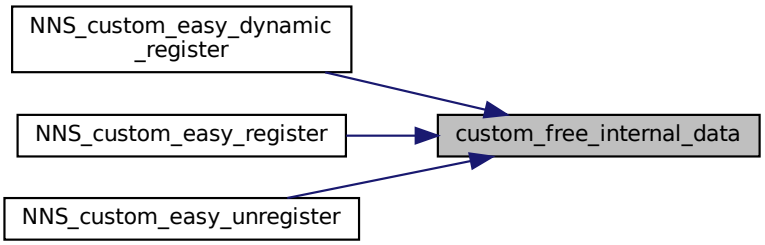
Internal function to release internal data.

Definition at line 62 of file tensor_filter_custom_easy.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.109.2.4 custom_getFrameworkInfo()

```
static int custom_getFrameworkInfo (
            const GstTensorFilterFramework * self,
            const GstTensorFilterProperties * prop,
            void * private_data,
            GstTensorFilterFrameworkInfo * fw_info )  [static]
```

V1 tensor-filter wrapper callback function, "getFrameworkInfo".

Definition at line 278 of file tensor_filter_custom_easy.c.

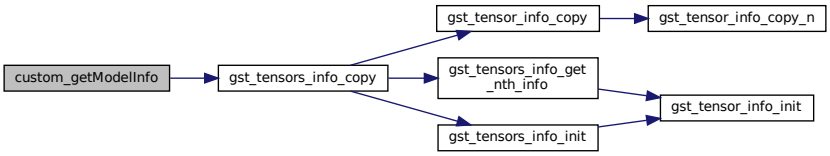
9.109.2.5 custom_getModelInfo()

```
static int custom_getModelInfo (
             const GstTensorFilterFramework * self,
             const GstTensorFilterProperties * prop,
             void * private_data,
             model_info_ops ops,
             GstTensorsInfo * in_info,
             GstTensorsInfo * out_info )  [static]
```

C V1 tensor-filter wrapper callback function, "getModelInfo".

Definition at line 300 of file tensor_filter_custom_easy.c.

Here is the call graph for this function:



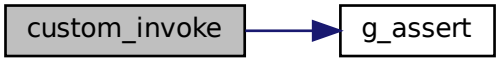
9.109.2.6 custom_invoke()

```
static int custom_invoke (
             const GstTensorFilterFramework * self,
             GstTensorFilterProperties * prop,
             void * private_data,
             const GstTensorMemory * input,
             GstTensorMemory * output )  [static]
```

Callback required by tensor_filter subplugin.

Definition at line 243 of file tensor_filter_custom_easy.c.

Here is the call graph for this function:



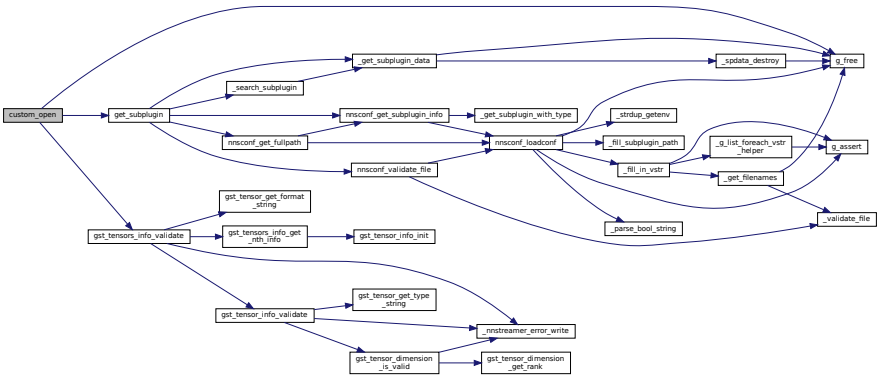
9.109.2.7 custom_open()

```
static int custom_open (
            const GstTensorFilterProperties * prop,
            void ** private_data )  [static]
```

Callback required by tensor_filter subplugin.

Definition at line 165 of file tensor_filter_custom_easy.c.

Here is the call graph for this function:



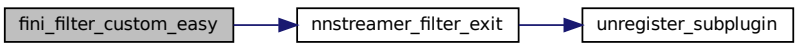
9.109.2.8 fini_filter_custom_easy()

```
void fini_filter_custom_easy (
            void  )
```

Destruct the subplugin.

Definition at line 352 of file tensor_filter_custom_easy.c.

Here is the call graph for this function:



9.109.2.9 init_filter_custom_easy()

```
void init_filter_custom_easy (
            void  )
```

internal_data

Initialize this object for tensor_filter subplugin runtime register. < The easy-filter writer's data

Definition at line 33 of file tensor_filter_custom_easy.c.

9.109.2.10 NNS_custom_easy_dynamic_register()

```
int NNS_custom_easy_dynamic_register (
            const char * modelname,
            NNS_custom_invoke_dynamic func,
            void * data,
            const GstTensorsInfo * in_info )
```

Register the custom-easy tensor function. More info in .h.

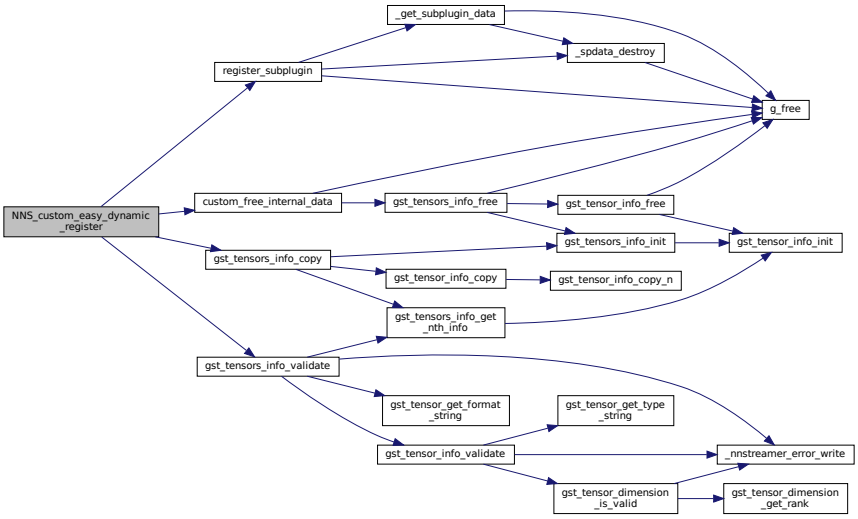
Register the custom-easy tensor function for dynamic invoke.

Returns

0 if success. -ERRNO if error.

Definition at line 112 of file tensor_filter_custom_easy.c.

Here is the call graph for this function:



9.109.2.11 NNS_custom_easy_register()

```
int NNS_custom_easy_register (
            const char * modelname,
            NNS_custom_invoke func,
            void * data,
            const GstTensorsInfo * in_info,
            const GstTensorsInfo * out_info )
```

Register the custom-easy tensor function. More info in .h.

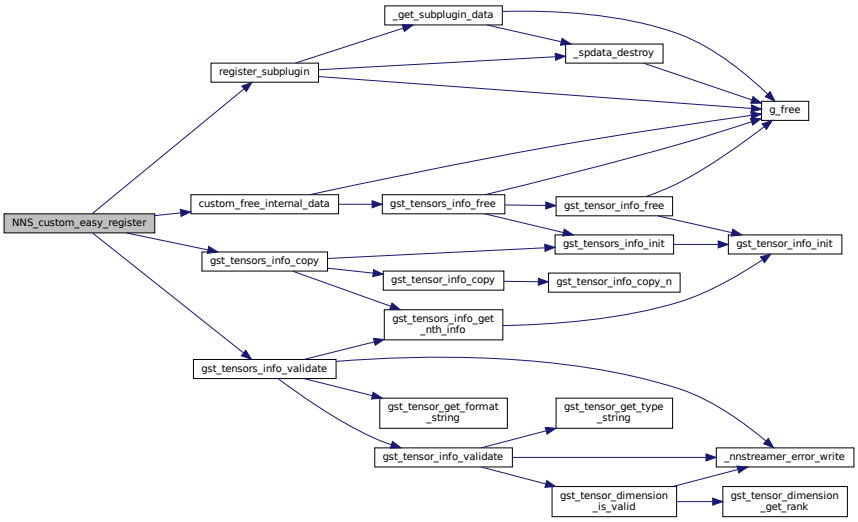
Register the custom-easy tensor function.

Returns

0 if success. -ERRNO if error.

Definition at line 76 of file tensor_filter_custom_easy.c.

Here is the call graph for this function:



9.109.2.12 NNS_custom_easy_unregister()

```
int NNS_custom_easy_unregister (
            const char * modelname )
```

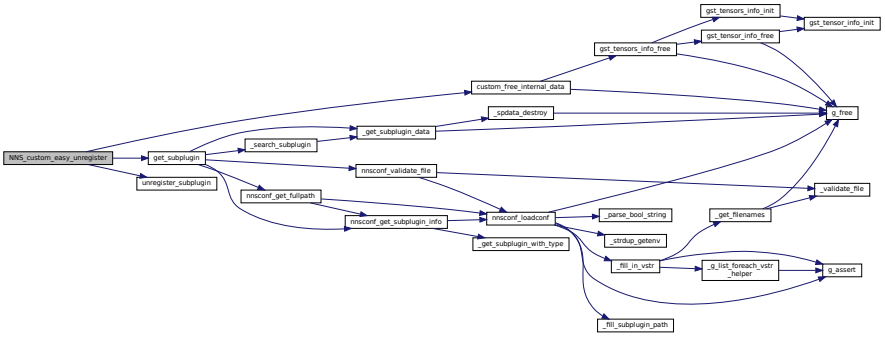
Unregister the custom-easy tensor function.

Returns

0 if success. -EINVAL if invalid model name.

Definition at line 144 of file tensor_filter_custom_easy.c.

Here is the call graph for this function:



9.109.3 Variable Documentation

9.109.3.1 internal_data

```
internal_data
```

Definition at line 48 of file tensor_filter_custom_easy.c.

9.109.3.2 NNS_support_custom_easy

```
GstTensorFilterFramework NNS_support_custom_easy  [static]
```

Initial value:

```
= {
  .version = GST_TENSOR_FILTER_FRAMEWORK_V1,
  .open = custom_open,
  .close = custom_close,
  .invoke = custom_invoke,
  .getFrameworkInfo = custom_getFrameworkInfo,
  .getModelInfo = custom_getModelInfo,
  .eventHandler = custom_eventHandler,
  .subplugin_data = NULL,
}
```

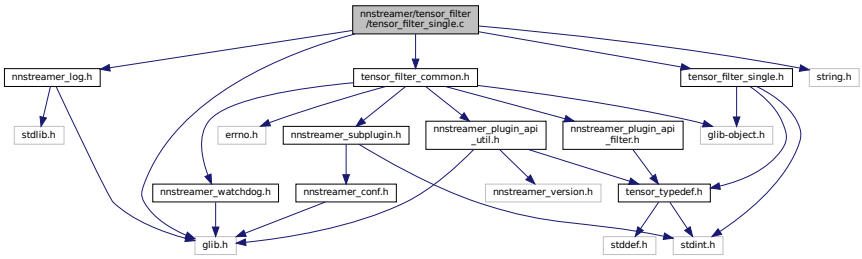
Definition at line 332 of file tensor_filter_custom_easy.c.

9.110 nnstreamer/tensor_filter/tensor_filter_single.c File Reference

Element to use general neural network framework directly without gstreamer pipeline.

```
#include <glib.h>
#include <string.h>
#include "nnstreamer_log.h"
#include "tensor_filter_common.h"
#include "tensor_filter_single.h"
```

Include dependency graph for tensor_filter_single.c:



Classes

- struct _GTensorFilterSinglePrivate

  Private data struct for tensor-filter single class.

Macros

- #define G_TENSOR_FILTER_SINGLE_PRIV(obj) ((GTensorFilterSinglePrivate *) (obj)->priv)
- #define g_tensor_filter_single_parent_class parent_class

Typedefs

- typedef struct _GTensorFilterSinglePrivate GTensorFilterSinglePrivate

  Private data struct for tensor-filter single class.

Functions

- G_DEFINE_TYPE_WITH_PRIVATE (GTensorFilterSingle, g_tensor_filter_single, G_TYPE_OBJECT)
- static void g_tensor_filter_single_finalize (GObject *object)

  Function to finalize instance.
- static void g_tensor_filter_single_set_property (GObject *object, guint prop_id, const GValue *value, G↩ ParamSpec *pspec)

  Setter for tensor_filter_single properties.
- static void g_tensor_filter_single_get_property (GObject *object, guint prop_id, GValue *value, GParamSpec *pspec)

  Getter for tensor_filter_single properties.

- static gboolean g_tensor_filter_single_invoke (GTensorFilterSingle *self, const GstTensorMemory *input, GstTensorMemory *output, gboolean allocate)

  Called when an input supposed to be invoked.
- static gboolean g_tensor_filter_input_configured (GTensorFilterSingle *self)

  Determine if input is configured (both input and output tensor)
- static gboolean g_tensor_filter_output_configured (GTensorFilterSingle *self)

  Determine if output is configured (both input and output tensor)
- static gint g_tensor_filter_set_input_info (GTensorFilterSingle *self, const GstTensorsInfo *in_info, GstTensorsInfo *out_info)

  Set input tensor information in the framework.
- static void g_tensor_filter_destroy_notify (GTensorFilterSingle *self, GstTensorMemory *mem)

  Called to notify the framework to destroy the allocated memory.
- static gboolean g_tensor_filter_allocate_in_invoke (GTensorFilterSingle *self)

  Determine if this filter framework supports allocation in invoke.
- static gboolean g_tensor_filter_single_start (GTensorFilterSingle *self)

  Called when the element starts processing, if fw not loaded.
- static gboolean g_tensor_filter_single_stop (GTensorFilterSingle *self)

  Called when the element stops processing, if fw loaded.
- static void g_tensor_filter_single_class_init (GTensorFilterSingleClass *klass)

  initialize the tensor_filter's class
- static void g_tensor_filter_single_init (GTensorFilterSingle *self)

  initialize the new element

9.110.1 Detailed Description

Element to use general neural network framework directly without gstreamer pipeline.

Date

28 Aug 2019

See also

http://github.com/nnstreamer/nnstreamer

Author

Parichay Kapoor pk.kapoor@samsung.com

Bug No known bugs except for NYI items

This is the main element for per-NN-framework plugins. Specific implementations for each NN framework must be written in each framework specific files; e.g., tensor_filter_tensorflow_lite.c

9.110.2 Macro Definition Documentation

9.110.2.1 g_tensor_filter_single_parent_class

```
#define g_tensor_filter_single_parent_class parent_class
```

Definition at line 61 of file tensor_filter_single.c.

9.110.2.2 G_TENSOR_FILTER_SINGLE_PRIV

```
#define G_TENSOR_FILTER_SINGLE_PRIV(
              obj ) ((GTensorFilterSinglePrivate *) (obj)->priv)
```

Definition at line 59 of file tensor_filter_single.c.

9.110.3 Typedef Documentation

9.110.3.1 GTensorFilterSinglePrivate

```
typedef struct _GTensorFilterSinglePrivate GTensorFilterSinglePrivate
```

Private data struct for tensor-filter single class.

SECTION:element-tensor_filter_single

An element that invokes neural network models and their framework or an independent shared object implementing tensor_filter_custom.h. The input and output are always in the format of other/tensor or other/tensors. This element is going to be the basis of single shot api.

9.110.4 Function Documentation

9.110.4.1 G_DEFINE_TYPE_WITH_PRIVATE()

```
G_DEFINE_TYPE_WITH_PRIVATE (
             GTensorFilterSingle ,
             g_tensor_filter_single ,
             G_TYPE_OBJECT  )
```

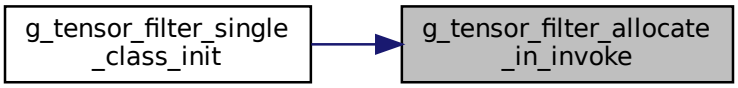
9.110.4.2 g_tensor_filter_allocate_in_invoke()

```
static gboolean g_tensor_filter_allocate_in_invoke (
            GTensorFilterSingle * self )  [inline], [static]
```

Determine if this filter framework supports allocation in invoke.

Definition at line 231 of file tensor_filter_single.c.

Here is the caller graph for this function:



9.110.4.3 g_tensor_filter_destroy_notify()

```
static void g_tensor_filter_destroy_notify (
            GTensorFilterSingle * self,
            GstTensorMemory * mem )  [static]
```

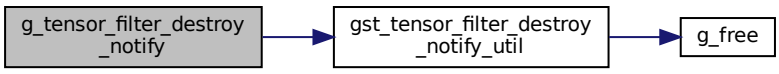
Called to notify the framework to destroy the allocated memory.

Parameters

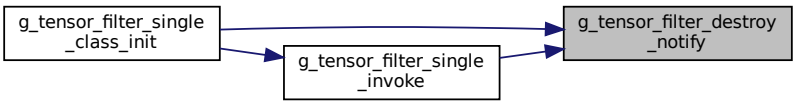
self	"this" pointer
mem	Memory wrapper for the allocated memory by the filter

Definition at line 296 of file tensor_filter_single.c.

Here is the call graph for this function:



Here is the caller graph for this function:



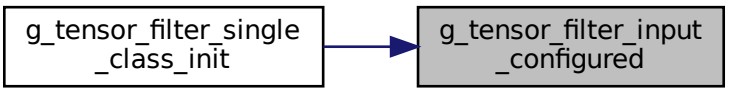
9.110.4.4 g_tensor_filter_input_configured()

```
static gboolean g_tensor_filter_input_configured (
            GTensorFilterSingle * self )  [static]
```

Determine if input is configured (both input and output tensor)

Definition at line 200 of file tensor_filter_single.c.

Here is the caller graph for this function:



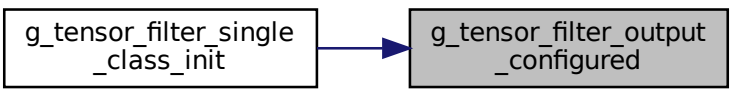
9.110.4.5 g_tensor_filter_output_configured()

```
static gboolean g_tensor_filter_output_configured (
            GTensorFilterSingle * self )  [static]
```

Determine if output is configured (both input and output tensor)

Definition at line 216 of file tensor_filter_single.c.

Here is the caller graph for this function:



9.110.4.6 g_tensor_filter_set_input_info()

```
static gint g_tensor_filter_set_input_info (
            GTensorFilterSingle * self,
            const GstTensorsInfo * in_info,
            GstTensorsInfo * out_info )  [static]
```

Set input tensor information in the framework.

Parameters

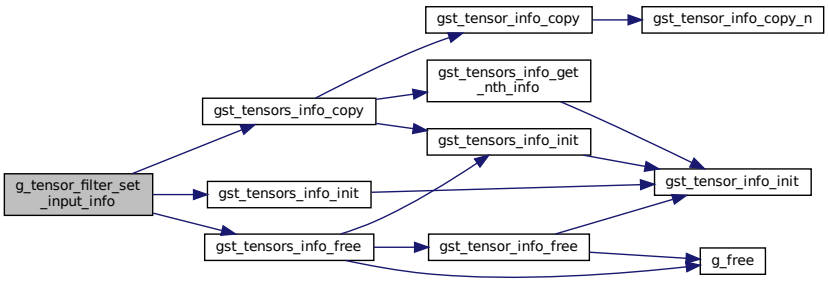
self	"this" pointer
in_info	information on the input tensor
out_info	updated information on the output tensor

Returns

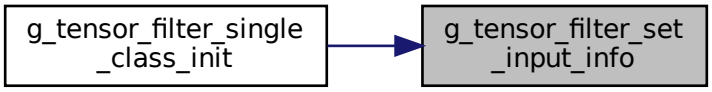
0 for success, -errno for failure.

Definition at line 400 of file tensor_filter_single.c.

Here is the call graph for this function:



Here is the caller graph for this function:



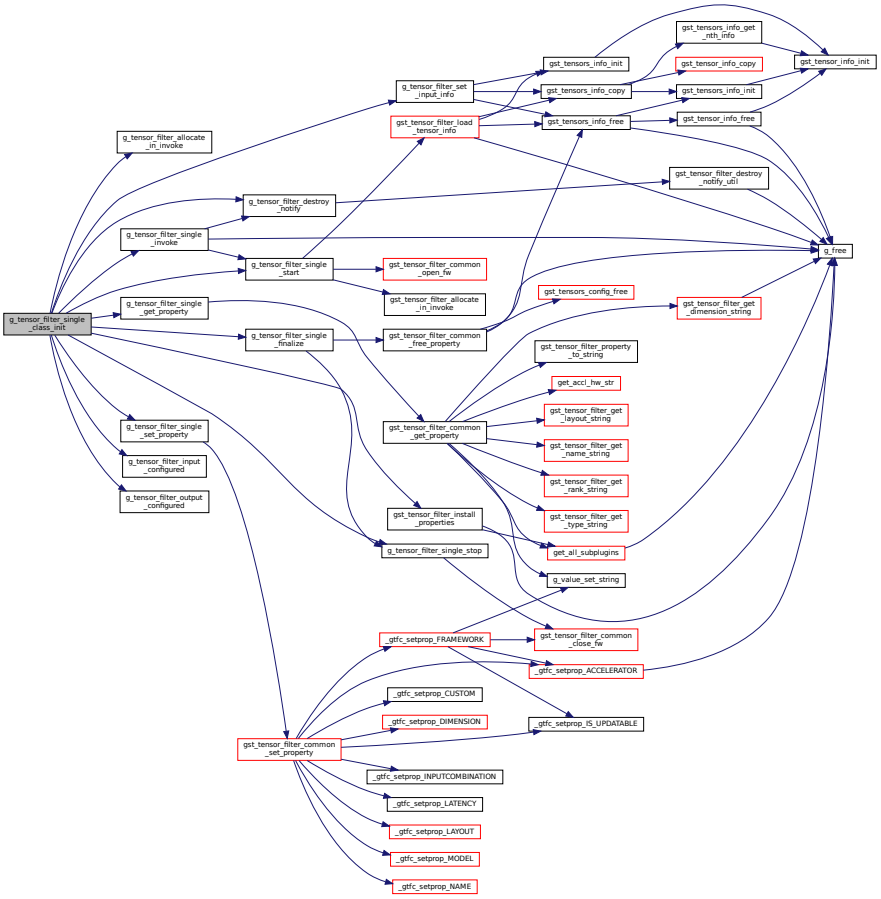
9.110.4.7 g_tensor_filter_single_class_init()

```
static void g_tensor_filter_single_class_init (
            GTensorFilterSingleClass * klass )  [static]
```

initialize the tensor_filter's class

Definition at line 89 of file tensor_filter_single.c.

Here is the call graph for this function:



9.110.4.8 g_tensor_filter_single_finalize()

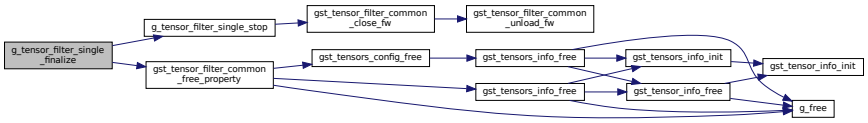
```
static void g_tensor_filter_single_finalize (
            GObject * object )  [static]
```

Function to finalize instance.

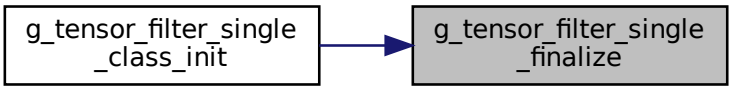
stop if not already stopped

Definition at line 133 of file tensor_filter_single.c.

Here is the call graph for this function:



Here is the caller graph for this function:



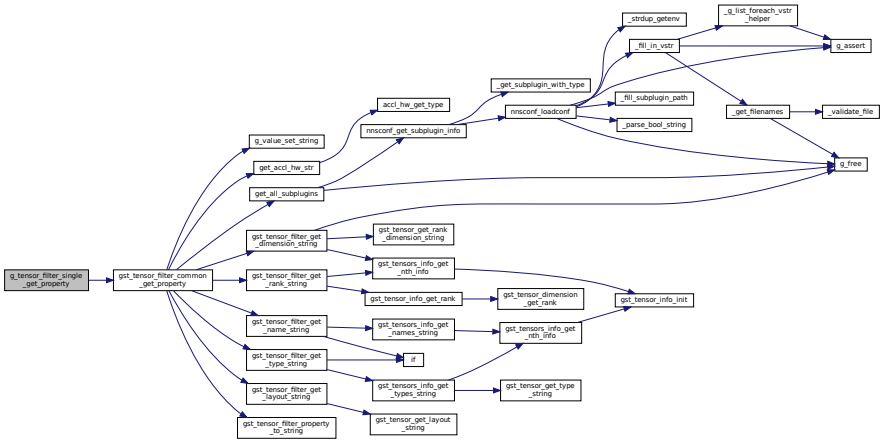
9.110.4.9 g_tensor_filter_single_get_property()

```
static void g_tensor_filter_single_get_property (
            GObject * object,
            guint prop_id,
            GValue * value,
            GParamSpec * pspec )  [static]
```

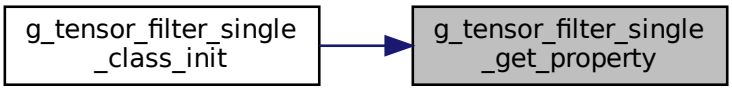
Getter for tensor_filter_single properties.

Definition at line 178 of file tensor_filter_single.c.

Here is the call graph for this function:



Here is the caller graph for this function:



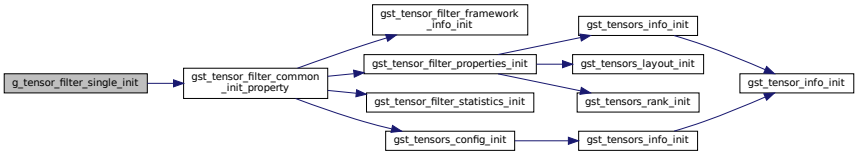
9.110.4.10 g_tensor_filter_single_init()

```
static void g_tensor_filter_single_init (
            GTensorFilterSingle * self )  [static]
```

initialize the new element

Definition at line 115 of file tensor_filter_single.c.

Here is the call graph for this function:



9.110.4.11 g_tensor_filter_single_invoke()

```
static gboolean g_tensor_filter_single_invoke (
            GTensorFilterSingle * self,
            const GstTensorMemory * input,
            GstTensorMemory * output,
            gboolean allocate )  [static]
```

Called when an input supposed to be invoked.

Parameters

self	"this" pointer
input	memory containing input data to run processing on
output	memory to put output data into after processing
allocate	true to allocate output data (false means tensor data is already allocated)

Returns

TRUE if there is no error.

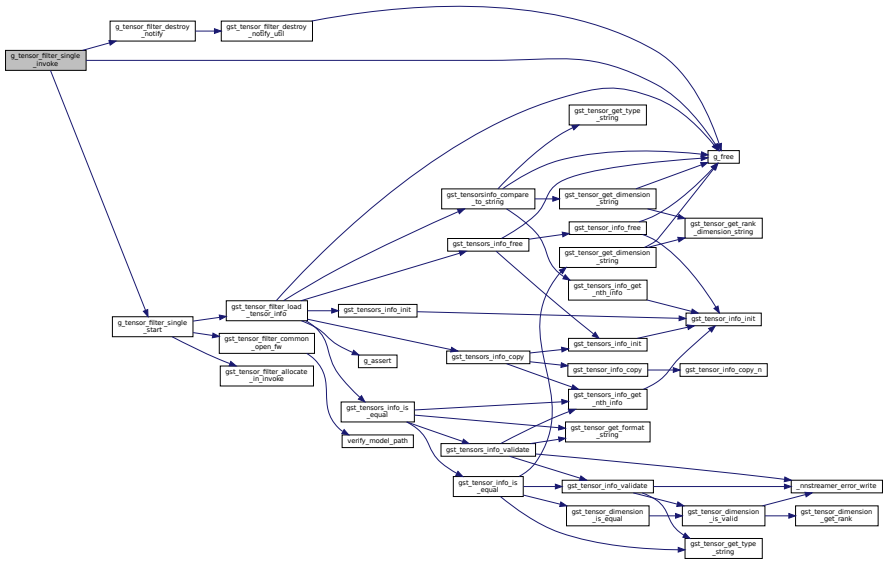
Todo refactor this local variable

start if not already started

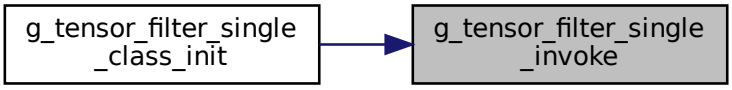
Todo how can we remove memcpy if output data is already allocated single-shot should fill the output data, but sub-plugin allocates new memory.

Definition at line 321 of file tensor_filter_single.c.

Here is the call graph for this function:



Here is the caller graph for this function:



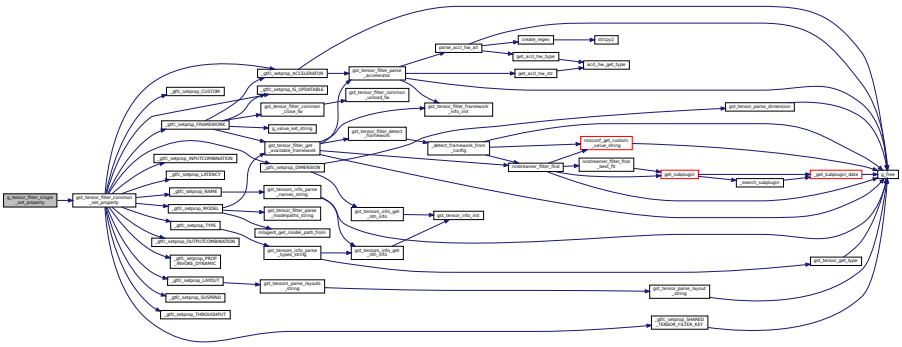
9.110.4.12 g_tensor_filter_single_set_property()

```
static void g_tensor_filter_single_set_property (
            GObject * object,
            guint prop_id,
            const GValue * value,
            GParamSpec * pspec )  [static]
```

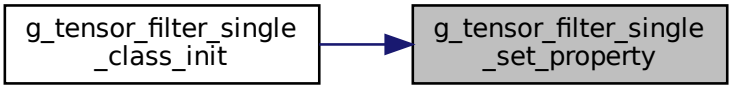
Setter for tensor_filter_single properties.

Definition at line 157 of file tensor_filter_single.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.110.4.13 g_tensor_filter_single_start()

```
static gboolean g_tensor_filter_single_start (
            GTensorFilterSingle * self )  [static]
```

Called when the element starts processing, if fw not loaded.

Parameters

self	"this" pointer

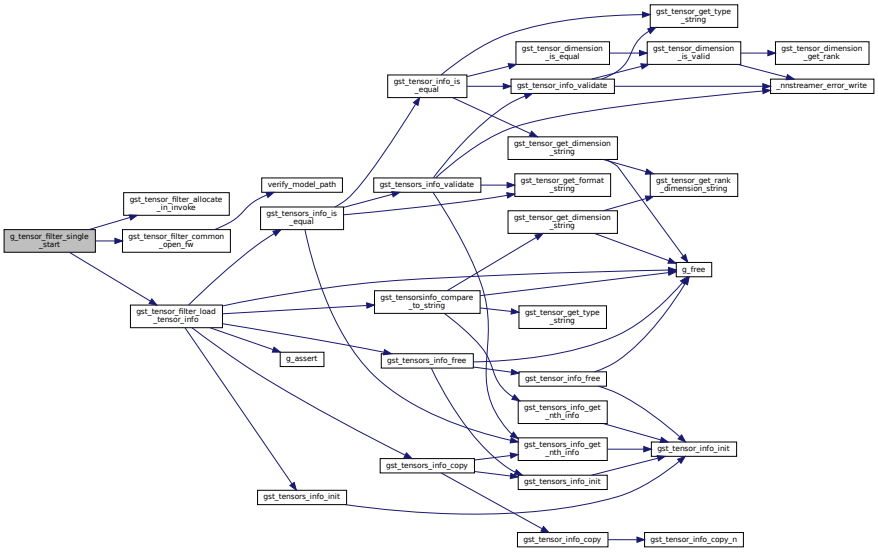
Returns

TRUE if there is no error.

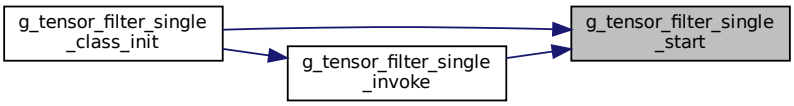
open framework, load model

Definition at line 246 of file tensor_filter_single.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.110.4.14 g_tensor_filter_single_stop()

```
static gboolean g_tensor_filter_single_stop (
            GTensorFilterSingle * self )  [static]
```

Called when the element stops processing, if fw loaded.

Parameters

self	"this" pointer

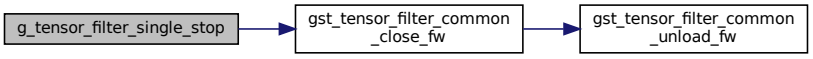
Returns

TRUE if there is no error.

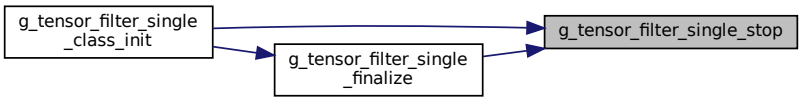
close framework, unload model

Definition at line 277 of file tensor_filter_single.c.

Here is the call graph for this function:



Here is the caller graph for this function:

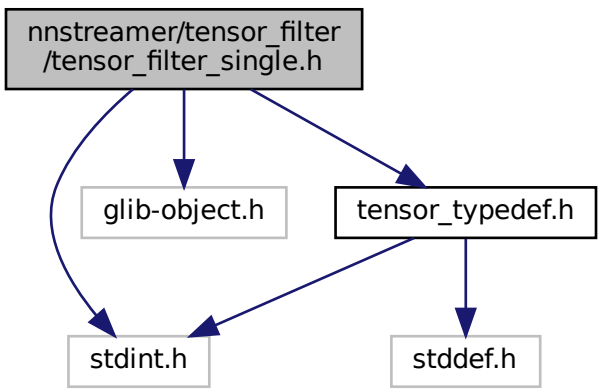


9.111 nnstreamer/tensor_filter/tensor_filter_single.h File Reference

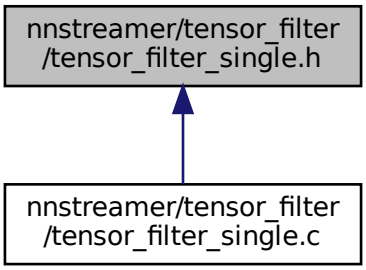
Element to use general neural network framework individually without gstreamer pipeline.

```
#include <stdint.h>
#include <glib-object.h>
#include <tensor_typedef.h>
```

Include dependency graph for tensor_filter_single.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct _GTensorFilterSingle

  Internal data structure for tensor_filter_single instances.
- struct _GTensorFilterSingleClass

  GTensorFilterSingleClass inherits GObjectClass.

Macros

- #define G_TYPE_TENSOR_FILTER_SINGLE (g_tensor_filter_single_get_type())
- #define G_TENSOR_FILTER_SINGLE(obj) (G_TYPE_CHECK_INSTANCE_CAST((obj),G_TYPE_TENSOR_FILTER_SINGLE
- #define G_TENSOR_FILTER_SINGLE_CLASS(klass) (G_TYPE_CHECK_CLASS_CAST((klass),G_TYPE_TENSOR_FILTER
- #define G_IS_TENSOR_FILTER_SINGLE(obj) (G_TYPE_CHECK_INSTANCE_TYPE((obj),G_TYPE_TENSOR_FILTER_SIN
- #define G_IS_TENSOR_FILTER_SINGLE_CLASS(klass) (G_TYPE_CHECK_CLASS_TYPE((klass),G_TYPE_TENSOR_FILT
- #define G_TENSOR_FILTER_SINGLE_CAST(obj) ((GTensorFilterSingle *)(obj))

Typedefs

- typedef struct _GTensorFilterSingle GTensorFilterSingle
- typedef struct _GTensorFilterSingleClass GTensorFilterSingleClass

Functions

- GType g_tensor_filter_single_get_type (void)

  Get Type function required for gst elements.

9.111.1 Detailed Description

Element to use general neural network framework individually without gstreamer pipeline.

Date

28 Aug 2019

See also

http://github.com/nnstreamer/nnstreamer

Author

Parichay Kapoor pk.kapoor@samsung.com
MyungJoo Ham myungjoo.ham@samsung.com

Bug No known bugs except for NYI items

9.111.2 Macro Definition Documentation

9.111.2.1 G_IS_TENSOR_FILTER_SINGLE

```
#define G_IS_TENSOR_FILTER_SINGLE(
              obj ) (G_TYPE_CHECK_INSTANCE_TYPE((obj),G_TYPE_TENSOR_FILTER_SINGLE))
```

Definition at line 39 of file tensor_filter_single.h.

9.111.2.2 G_IS_TENSOR_FILTER_SINGLE_CLASS

```
#define G_IS_TENSOR_FILTER_SINGLE_CLASS(
              klass ) (G_TYPE_CHECK_CLASS_TYPE((klass),G_TYPE_TENSOR_FILTER_SINGLE))
```

Definition at line 41 of file tensor_filter_single.h.

9.111.2.3 G_TENSOR_FILTER_SINGLE

```
#define G_TENSOR_FILTER_SINGLE(
              obj ) (G_TYPE_CHECK_INSTANCE_CAST((obj),G_TYPE_TENSOR_FILTER_SINGLE,GTensorFilterSingle))
```

Definition at line 35 of file tensor_filter_single.h.

9.111.2.4 G_TENSOR_FILTER_SINGLE_CAST

```
#define G_TENSOR_FILTER_SINGLE_CAST(
              obj ) ((GTensorFilterSingle *)(obj))
```

Definition at line 43 of file tensor_filter_single.h.

9.111.2.5 G_TENSOR_FILTER_SINGLE_CLASS

```
#define G_TENSOR_FILTER_SINGLE_CLASS(
              klass ) (G_TYPE_CHECK_CLASS_CAST((klass),G_TYPE_TENSOR_FILTER_SINGLE,GTensorFilterSingleClass))
```

Definition at line 37 of file tensor_filter_single.h.

9.111.2.6 G_TYPE_TENSOR_FILTER_SINGLE

```
#define G_TYPE_TENSOR_FILTER_SINGLE (g_tensor_filter_single_get_type())
```

Definition at line 33 of file tensor_filter_single.h.

9.111.3 Typedef Documentation

9.111.3.1 GTensorFilterSingle

```
typedef struct _GTensorFilterSingle GTensorFilterSingle
```

Definition at line 45 of file tensor_filter_single.h.

9.111.3.2 GTensorFilterSingleClass

```
typedef struct _GTensorFilterSingleClass GTensorFilterSingleClass
```

Definition at line 46 of file tensor_filter_single.h.

9.111.4 Function Documentation

9.111.4.1 g_tensor_filter_single_get_type()

```
GType g_tensor_filter_single_get_type (
            void  )
```

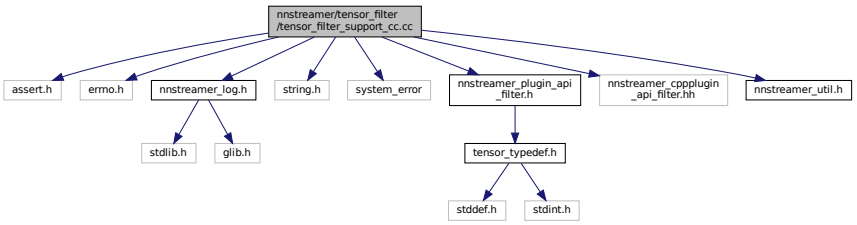
Get Type function required for gst elements.

9.112 nnstreamer/tensor_filter/tensor_filter_support_cc.cc File Reference

Base class for tensor_filter subplugins of C++ classes.

```
#include <assert.h>
#include <errno.h>
#include <nnstreamer_log.h>
#include <string.h>
#include <system_error>
#include <nnstreamer_plugin_api_filter.h>
#include <nnstreamer_cpppplugin_api_filter.hh>
#include <nnstreamer_util.h>
```

Include dependency graph for tensor_filter_support_cc.cc:



Namespaces

- nnstreamer

Macros

- #define NO_ANONYMOUS_NESTED_STRUCT
- #define _SANITY_CHECK (0xFACE217714DEADE7ULL)
- #define _RETURN_ERR_WITH_MSG(c, m)
- #define GET_TFSP_WITH_CHECKS(obj, private_data)

9.112.1 Detailed Description

Base class for tensor_filter subplugins of C++ classes.

GStreamer Tensor_filter, C++ Subplugin Support. (this is not a subplugin) Copyright (C) 2020 MyungJoo Ham myungjoo.ham@samsung.com

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Library General Public License as published by the Free Software Foundation; version 2.1 of the License.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Library General Public License for more details.

Date

22 Jan 2020

See also

http://github.com/nnstreamer/nnstreamer

Author

MyungJoo Ham myungjoo.ham@samsung.com

Bug No known bugs except for NYI items

This is not a subplugin, but a helper for C++ subplugins. If you want to write a wrapper for neural network frameworks or a hardware adaptor in C++, this is what you want. If you want to attach a single C++ class/object as a filter (a.k.a. custom filter) to a pipeline, use tensor_filter_cpp

9.112.2 Macro Definition Documentation

9.112.2.1 _RETURN_ERR_WITH_MSG

```
#define _RETURN_ERR_WITH_MSG(
              c,
              m )
```

Value:

```
  do {                                \
    nns_loge ("%s", m);               \
    return c;                         \
  } while (0);
```

Definition at line 54 of file tensor_filter_support_cc.cc.

9.112.2.2 _SANITY_CHECK

```
#define _SANITY_CHECK (0xFACE217714DEADE7ULL)
```

Definition at line 53 of file tensor_filter_support_cc.cc.

9.112.2.3 GET_TFSP_WITH_CHECKS

```
#define GET_TFSP_WITH_CHECKS(
              obj,
              private_data )
```

Value:

```
  do {                                                                  \
    try {                                                               \
      obj = get_tfsp_with_checks (private_data);                        \
    } catch (const std::exception &e) {                                 \                          \
      return -EINVAL;                                                   \ \
    }                                                                   \
  } while (0);
```

Definition at line 60 of file tensor_filter_support_cc.cc.

9.112.2.4 NO_ANONYMOUS_NESTED_STRUCT

```
#define NO_ANONYMOUS_NESTED_STRUCT
```

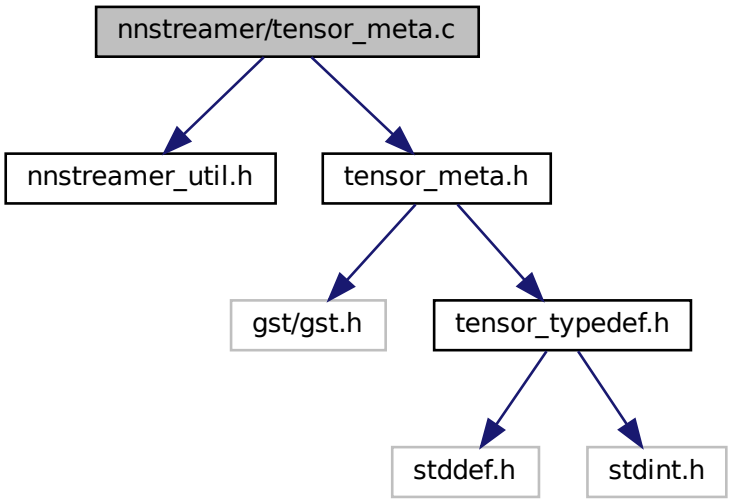
Definition at line 39 of file tensor_filter_support_cc.cc.

9.113 nnstreamer/tensor_meta.c File Reference

Internal tensor meta implementation for nnstreamer.

```
#include <nnstreamer_util.h>
#include "tensor_meta.h"
```

Include dependency graph for tensor_meta.c:



Functions

- GType gst_meta_query_api_get_type (void)
  Define meta_query type to register.
- static gboolean gst_meta_query_init (GstMeta *meta, gpointer params, GstBuffer *buffer)
  meta_query init
- static void gst_meta_query_free (GstMeta *meta, GstBuffer *buffer)
  free meta_query
- static gboolean gst_meta_query_transform (GstBuffer *transbuf, GstMeta *meta, GstBuffer *buffer, GQuark type, gpointer data)
  tensor_query meta data transform (source to dest)
- const GstMetaInfo * gst_meta_query_get_info (void)
  Get meta_query info.

9.113.1 Detailed Description

Internal tensor meta implementation for nnstreamer.

Copyright (C) 2021 Junhwan Kim jejudo.kim@samsung.com

Date

09 Aug 2021

Author

Junhwan Kim jejudo.kim@samsung.com

See also

http://github.com/nnstreamer/nnstreamer

Bug No known bugs

9.113.2 Function Documentation

9.113.2.1 gst_meta_query_api_get_type()

```
GType gst_meta_query_api_get_type (
            void  )
```

Define meta_query type to register.

Definition at line 24 of file tensor_meta.c.

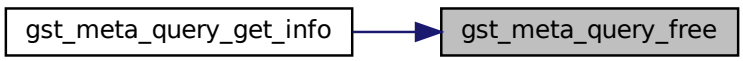
9.113.2.2 gst_meta_query_free()

```
static void gst_meta_query_free (
            GstMeta * meta,
            GstBuffer * buffer )  [static]
```

free meta_query

Definition at line 61 of file tensor_meta.c.

Here is the caller graph for this function:



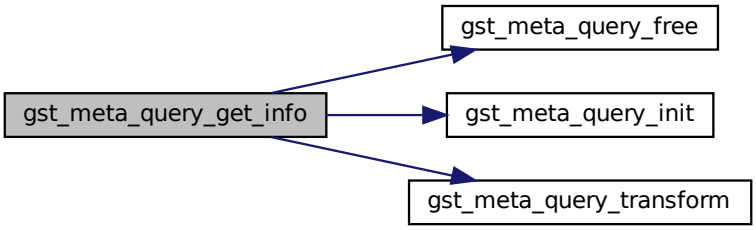
9.113.2.3 gst_meta_query_get_info()

```
const GstMetaInfo* gst_meta_query_get_info (
            void  )
```

Get meta_query info.

Definition at line 87 of file tensor_meta.c.

Here is the call graph for this function:



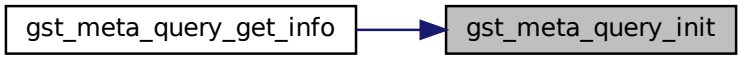
9.113.2.4 gst_meta_query_init()

```
static gboolean gst_meta_query_init (
            GstMeta * meta,
            gpointer params,
            GstBuffer * buffer )  [static]
```

meta_query init

Definition at line 48 of file tensor_meta.c.

Here is the caller graph for this function:



9.113.2.5 gst_meta_query_transform()

```
static gboolean gst_meta_query_transform (
            GstBuffer * transbuf,
            GstMeta * meta,
            GstBuffer * buffer,
            GQuark type,
            gpointer data )  [static]
```

tensor_query meta data transform (source to dest)

Definition at line 71 of file tensor_meta.c.

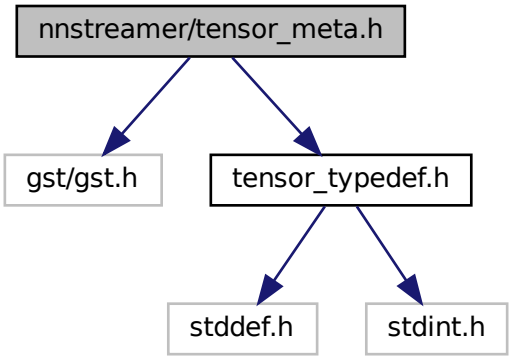
Here is the caller graph for this function:

9.114 nnstreamer/tensor_meta.h File Reference

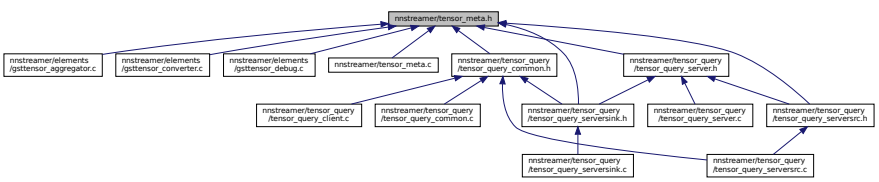
Internal tensor meta header for nnstreamer.

```
#include <gst/gst.h>
#include <tensor_typedef.h>
```

Include dependency graph for tensor_meta.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct GstMetaQuery

  GstMetaQuery meta structure.

Macros

- #define GST_META_QUERY_API_TYPE (gst_meta_query_api_get_type())
- #define GST_META_QUERY_INFO (gst_meta_query_get_info())
- #define gst_buffer_get_meta_query(b) ((GstMetaQuery *) gst_buffer_get_meta ((b), GST_META_QUERY_API_TYPE))
- #define gst_buffer_add_meta_query(b) ((GstMetaQuery *) gst_buffer_add_meta ((b), GST_META_QUERY_INFO, NULL))

Functions

- GType gst_meta_query_api_get_type (void)

  Define meta_query type to register.
- const GstMetaInfo * gst_meta_query_get_info (void)

  Get meta_query info.

Variables

- G_BEGIN_DECLS typedef int64_t query_client_id_t

9.114.1 Detailed Description

Internal tensor meta header for nnstreamer.

Date

09 Aug 2021

Author

Junhwan Kim jejudo.kim@samsung.com

See also

http://github.com/nnstreamer/nnstreamer

Bug No known bugs

9.114.2 Macro Definition Documentation

9.114.2.1 gst_buffer_add_meta_query

```
#define gst_buffer_add_meta_query(
              b ) ((GstMetaQuery *) gst_buffer_add_meta ((b), GST_META_QUERY_INFO, NULL))
```

Definition at line 46 of file tensor_meta.h.

9.114.2.2 gst_buffer_get_meta_query

```
#define gst_buffer_get_meta_query(
              b ) ((GstMetaQuery *) gst_buffer_get_meta ((b), GST_META_QUERY_API_TYPE))
```

Definition at line 44 of file tensor_meta.h.

9.114.2.3 GST_META_QUERY_API_TYPE

```
#define GST_META_QUERY_API_TYPE (gst_meta_query_api_get_type())
```

Definition at line 37 of file tensor_meta.h.

9.114.2.4 GST_META_QUERY_INFO

```
#define GST_META_QUERY_INFO (gst_meta_query_get_info())
```

Definition at line 43 of file tensor_meta.h.

9.114.3 Function Documentation

9.114.3.1 gst_meta_query_api_get_type()

```
GType gst_meta_query_api_get_type (
             void  )
```

Define meta_query type to register.

Definition at line 24 of file tensor_meta.c.

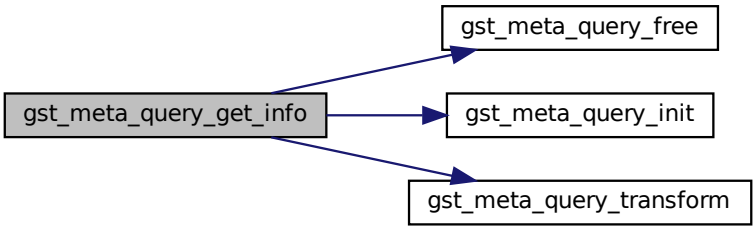
9.114.3.2 gst_meta_query_get_info()

```
const GstMetaInfo* gst_meta_query_get_info (
            void  )
```

Get meta_query info.

Definition at line 87 of file tensor_meta.c.

Here is the call graph for this function:



9.114.4 Variable Documentation

9.114.4.1 query_client_id_t

```
G_BEGIN_DECLS typedef int64_t query_client_id_t
```

Definition at line 21 of file tensor_meta.h.

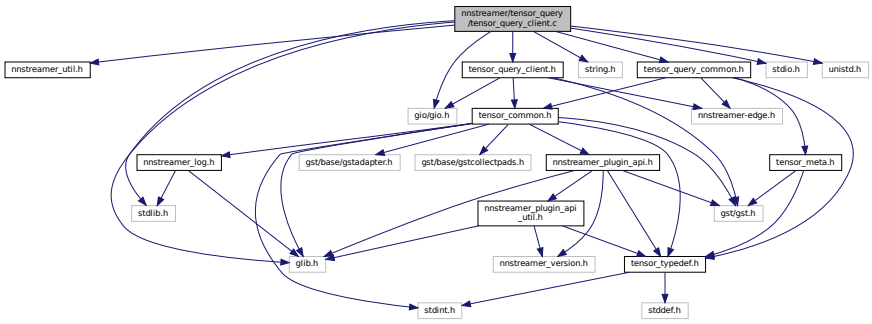
9.115 nnstreamer/tensor_query/tensor_query_client.c File Reference

GStreamer plugin to handle tensor query client.

```
#include "nnstreamer_util.h"
#include "tensor_query_client.h"
#include <gio/gio.h>
#include <glib.h>
#include <string.h>
#include "tensor_query_common.h"
#include <stdio.h>
#include <stdlib.h>
```

```
#include <unistd.h>
```

Include dependency graph for tensor_query_client.c:



Macros

- #define DBG (!self->silent)

  Macro for debug mode.
- #define TCP_HIGHEST_PORT 65535
- #define TCP_DEFAULT_HOST "localhost"
- #define TCP_DEFAULT_SRV_SRC_PORT 3000
- #define TCP_DEFAULT_CLIENT_SRC_PORT 3001
- #define DEFAULT_CLIENT_TIMEOUT 0
- #define DEFAULT_SILENT TRUE
- #define DEFAULT_MAX_REQUEST 2
- #define GST_CAT_DEFAULT gst_tensor_query_client_debug
- #define gst_tensor_query_client_parent_class parent_class

Enumerations

- enum {
  PROP_0, PROP_HOST, PROP_PORT, PROP_DEST_HOST,
  PROP_DEST_PORT, PROP_CONNECT_TYPE, PROP_TOPIC, PROP_TIMEOUT,
  PROP_SILENT, PROP_MAX_REQUEST }

  Properties.

Functions

- GST_DEBUG_CATEGORY_STATIC (gst_tensor_query_client_debug)
- G_DEFINE_TYPE (GstTensorQueryClient, gst_tensor_query_client, GST_TYPE_ELEMENT)
- static void gst_tensor_query_client_finalize (GObject *object)

  finalize the object
- static void gst_tensor_query_client_set_property (GObject *object, guint prop_id, const GValue *value, G↩ ParamSpec *pspec)

  set property
- static void gst_tensor_query_client_get_property (GObject *object, guint prop_id, GValue *value, GParam↩ Spec *pspec)

  get property
- static gboolean gst_tensor_query_client_sink_event (GstPad *pad, GstObject *parent, GstEvent *event)

This function handles sink event.

- static gboolean gst_tensor_query_client_sink_query (GstPad *pad, GstObject *parent, GstQuery *query)

  This function handles sink pad query.
- static GstFlowReturn gst_tensor_query_client_chain (GstPad *pad, GstObject *parent, GstBuffer *buf)

  Chain function, this function does the actual processing.
- static GstCaps * gst_tensor_query_client_query_caps (GstTensorQueryClient *self, GstPad *pad, GstCaps *filter)

  Get pad caps for caps negotiation.
- static void gst_tensor_query_client_class_init (GstTensorQueryClientClass *klass)

  initialize the class
- static void gst_tensor_query_client_init (GstTensorQueryClient *self)

  initialize the new element
- static gboolean gst_tensor_query_client_update_caps (GstTensorQueryClient *self, const gchar *caps_str)

  Update src pad caps from tensors config.
- static gchar * _nns_edge_parse_caps (gchar *caps_str, gboolean is_src)

  Parse caps from received event data.
- static int _nns_edge_event_cb (nns_edge_event_h event_h, void *user_data)

  nnstreamer-edge event callback.
- static gboolean gst_tensor_query_client_create_edge_handle (GstTensorQueryClient *self)

  Internal function to create edge handle.

Variables

- static GstStaticPadTemplate sinktemplate

  the capabilities of the inputs.
- static GstStaticPadTemplate srctemplate

  the capabilities of the outputs.

9.115.1 Detailed Description

GStreamer plugin to handle tensor query client.

Date

09 Jul 2021

Author

Junhwan Kim jejudo.kim@samsung.com

See also

http://github.com/nnstreamer/nnstreamer

Bug No known bugs

9.115.2 Macro Definition Documentation

9.115.2.1 DBG

```
#define DBG (!self->silent)
```

Macro for debug mode.

Definition at line 32 of file tensor_query_client.c.

9.115.2.2 DEFAULT_CLIENT_TIMEOUT

```
#define DEFAULT_CLIENT_TIMEOUT 0
```

Definition at line 56 of file tensor_query_client.c.

9.115.2.3 DEFAULT_MAX_REQUEST

```
#define DEFAULT_MAX_REQUEST 2
```

Definition at line 58 of file tensor_query_client.c.

9.115.2.4 DEFAULT_SILENT

```
#define DEFAULT_SILENT TRUE
```

Definition at line 57 of file tensor_query_client.c.

9.115.2.5 GST_CAT_DEFAULT

```
#define GST_CAT_DEFAULT gst_tensor_query_client_debug
```

Definition at line 61 of file tensor_query_client.c.

9.115.2.6 gst_tensor_query_client_parent_class

```
#define gst_tensor_query_client_parent_class parent_class
```

Definition at line 79 of file tensor_query_client.c.

9.115.2.7 TCP_DEFAULT_CLIENT_SRC_PORT

```
#define TCP_DEFAULT_CLIENT_SRC_PORT 3001
```

Definition at line 55 of file tensor_query_client.c.

9.115.2.8 TCP_DEFAULT_HOST

```
#define TCP_DEFAULT_HOST "localhost"
```

Definition at line 53 of file tensor_query_client.c.

9.115.2.9 TCP_DEFAULT_SRV_SRC_PORT

```
#define TCP_DEFAULT_SRV_SRC_PORT 3000
```

Definition at line 54 of file tensor_query_client.c.

9.115.2.10 TCP_HIGHEST_PORT

```
#define TCP_HIGHEST_PORT 65535
```

Definition at line 52 of file tensor_query_client.c.

9.115.3 Enumeration Type Documentation

9.115.3.1 anonymous enum

```
anonymous enum
```

Properties.

Enumerator

PROP_0	
PROP_HOST	
PROP_PORT	
PROP_DEST_HOST	
PROP_DEST_PORT	
PROP_CONNECT_TYPE	
PROP_TOPIC	
PROP_TIMEOUT	
PROP_SILENT	
PROP_MAX_REQUEST	

Definition at line 38 of file tensor_query_client.c.

9.115.4 Function Documentation

9.115.4.1 _nns_edge_event_cb()

```
static int _nns_edge_event_cb (
            nns_edge_event_h event_h,
            void * user_data )  [static]
```

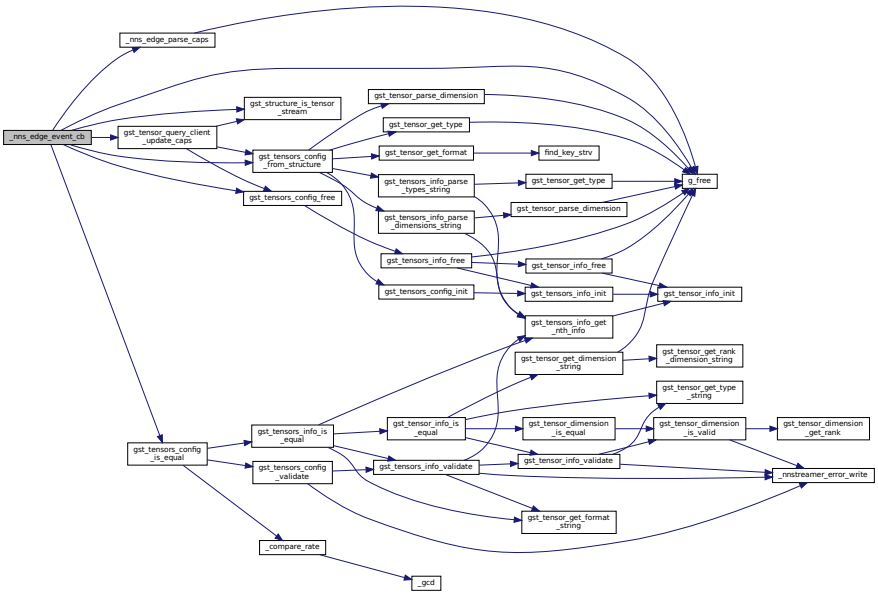
nnstreamer-edge event callback.

Server framerate may vary. Let's skip comparing the framerate.

Update client src caps

Definition at line 435 of file tensor_query_client.c.

Here is the call graph for this function:



Here is the caller graph for this function:

9.115.4.2 _nns_edge_parse_caps()

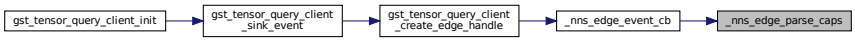
```
static gchar* _nns_edge_parse_caps (
            gchar * caps_str,
            gboolean is_src )  [static]
```

Parse caps from received event data.

Definition at line 400 of file tensor_query_client.c.

Here is the call graph for this function:

Here is the caller graph for this function:



9.115.4.3 G_DEFINE_TYPE()

```
G_DEFINE_TYPE (
            GstTensorQueryClient ,
            gst_tensor_query_client ,
            GST_TYPE_ELEMENT  )
```

9.115.4.4 GST_DEBUG_CATEGORY_STATIC()

```
GST_DEBUG_CATEGORY_STATIC (
            gst_tensor_query_client_debug  )
```

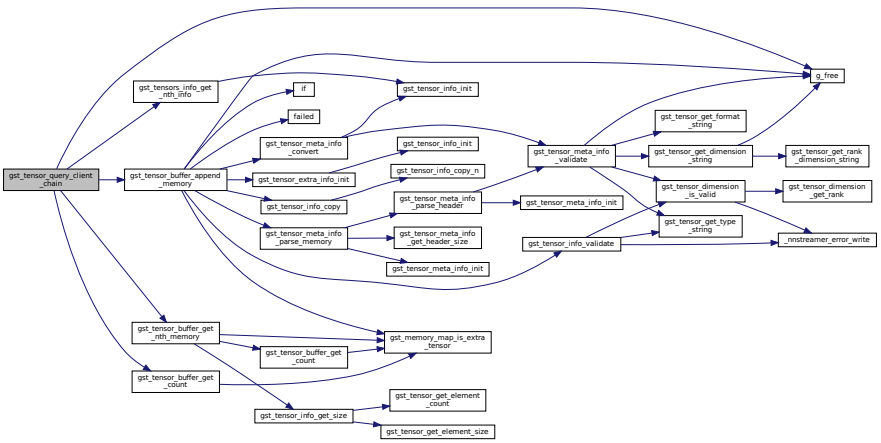
9.115.4.5 gst_tensor_query_client_chain()

```
static GstFlowReturn gst_tensor_query_client_chain (
            GstPad * pad,
            GstObject * parent,
            GstBuffer * buf )  [static]
```

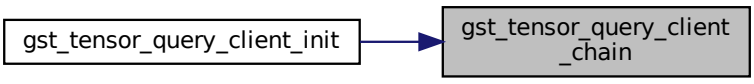
Chain function, this function does the actual processing.

Definition at line 674 of file tensor_query_client.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.115.4.6 gst_tensor_query_client_class_init()

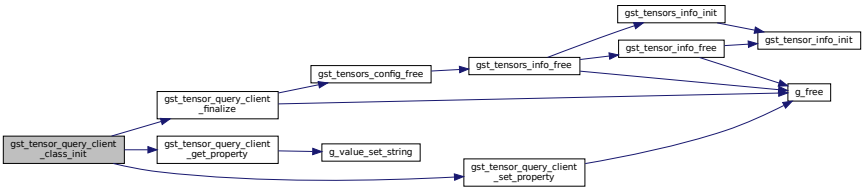
```
static void gst_tensor_query_client_class_init (
            GstTensorQueryClientClass * klass )  [static]
```

initialize the class

install property goes here

Definition at line 101 of file tensor_query_client.c.

Here is the call graph for this function:



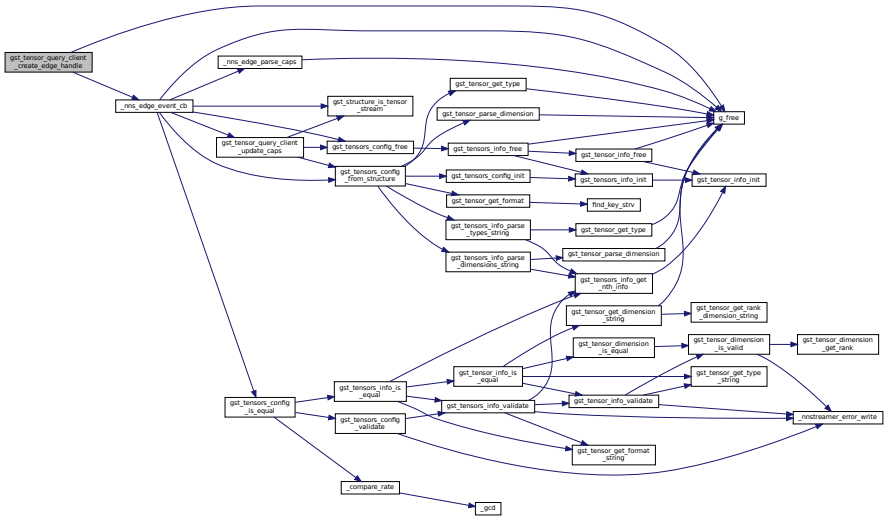
9.115.4.7 gst_tensor_query_client_create_edge_handle()

```
static gboolean gst_tensor_query_client_create_edge_handle (
            GstTensorQueryClient * self )  [static]
```

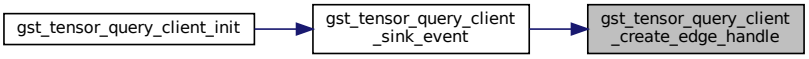
Internal function to create edge handle.

Definition at line 521 of file tensor_query_client.c.

Here is the call graph for this function:



Here is the caller graph for this function:



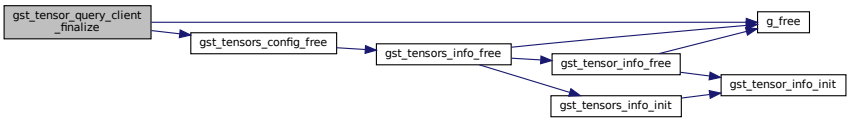
9.115.4.8 gst_tensor_query_client_finalize()

```
static void gst_tensor_query_client_finalize (
            GObject * object )  [static]
```

finalize the object

Definition at line 213 of file tensor_query_client.c.

Here is the call graph for this function:



Here is the caller graph for this function:

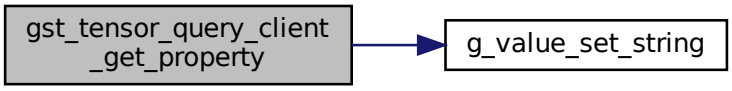
9.115.4.9 gst_tensor_query_client_get_property()

```
static void gst_tensor_query_client_get_property (
             GObject * object,
             guint prop_id,
             GValue * value,
             GParamSpec * pspec )  [static]
```

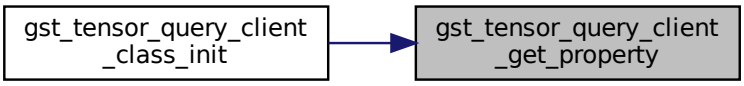
get property

Definition at line 309 of file tensor_query_client.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.115.4.10 gst_tensor_query_client_init()

```
static void gst_tensor_query_client_init (
             GstTensorQueryClient * self )  [static]
```

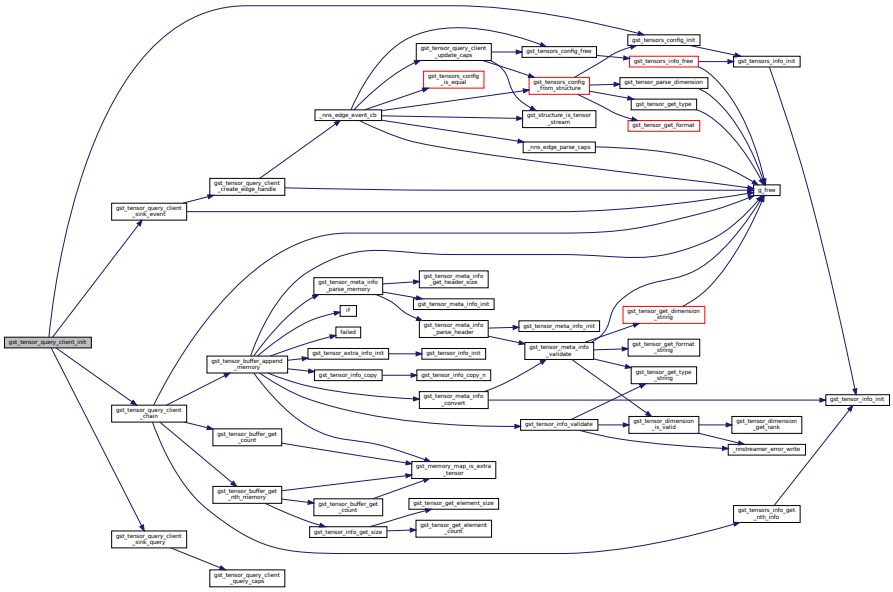
initialize the new element

setup sink pad

setup src pad

Definition at line 175 of file tensor_query_client.c.

Here is the call graph for this function:



9.115.4.11 gst_tensor_query_client_query_caps()

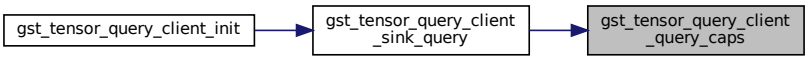
```
static GstCaps * gst_tensor_query_client_query_caps (
            GstTensorQueryClient * self,
            GstPad * pad,
            GstCaps * filter )  [static]
```

Get pad caps for caps negotiation.

pad don't have current caps. use the template caps

Definition at line 784 of file tensor_query_client.c.

Here is the caller graph for this function:



9.115.4.12 gst_tensor_query_client_set_property()

```
static void gst_tensor_query_client_set_property (
            GObject * object,
            guint prop_id,
            const GValue * value,
            GParamSpec * pspec )  [static]
```

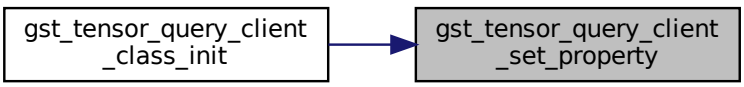
set property

Todo DO NOT update properties (host, port, ..) while pipeline is running.

Definition at line 250 of file tensor_query_client.c.

Here is the call graph for this function:

Here is the caller graph for this function:



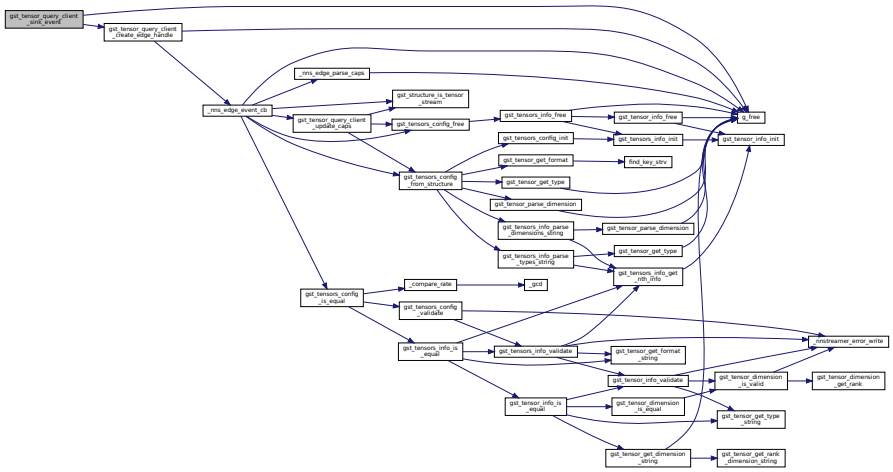
9.115.4.13 gst_tensor_query_client_sink_event()

```
static gboolean gst_tensor_query_client_sink_event (
            GstPad * pad,
            GstObject * parent,
            GstEvent * event )  [static]
```

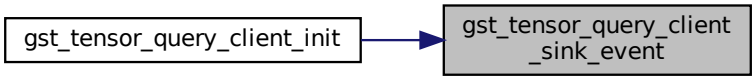
This function handles sink event.

Definition at line 587 of file tensor_query_client.c.

Here is the call graph for this function:



Here is the caller graph for this function:



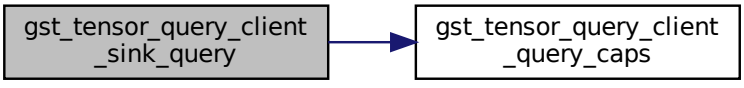
9.115.4.14 gst_tensor_query_client_sink_query()

```
static gboolean gst_tensor_query_client_sink_query (
            GstPad * pad,
            GstObject * parent,
            GstQuery * query )  [static]
```

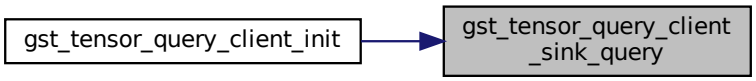
This function handles sink pad query.

Definition at line 623 of file tensor_query_client.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.115.4.15 gst_tensor_query_client_update_caps()

```
static gboolean gst_tensor_query_client_update_caps (
            GstTensorQueryClient * self,
            const gchar * caps_str )  [static]
```

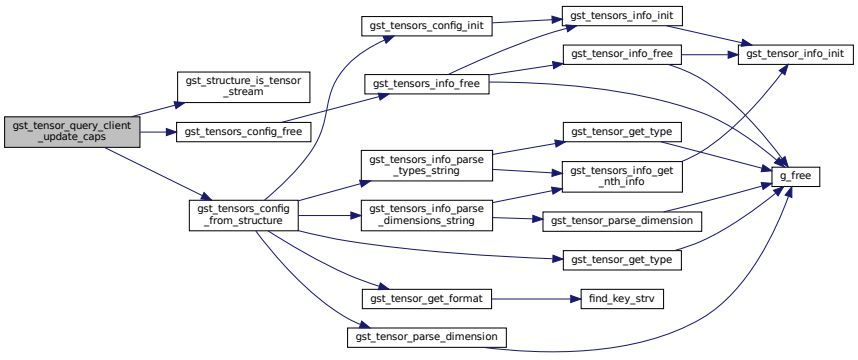
Update src pad caps from tensors config.

If pad caps is updated, prepare the tensor information here. It will be used in chain function, to push tensor buffer into src pad.

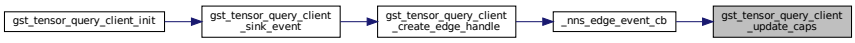
Don't need to update when the capability is the same.

Definition at line 352 of file tensor_query_client.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.115.5 Variable Documentation

9.115.5.1 sinktemplate

```
GstStaticPadTemplate sinktemplate  [static]
```

Initial value:

```
= GST_STATIC_PAD_TEMPLATE ("sink",
    GST_PAD_SINK,
    GST_PAD_ALWAYS,
    GST_STATIC_CAPS_ANY)
```

the capabilities of the inputs.

Definition at line 66 of file tensor_query_client.c.

9.115.5.2 srctemplate

```
GstStaticPadTemplate srctemplate  [static]
```

Initial value:

```
= GST_STATIC_PAD_TEMPLATE ("src",
    GST_PAD_SRC,
    GST_PAD_ALWAYS,
    GST_STATIC_CAPS_ANY)
```

the capabilities of the outputs.

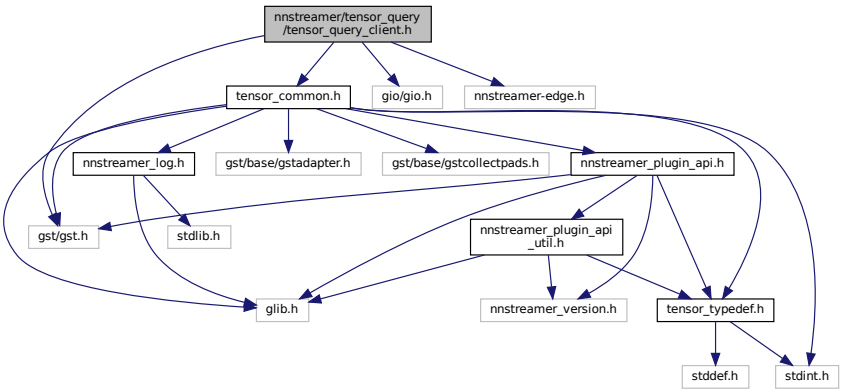
Definition at line 74 of file tensor_query_client.c.

9.116 nnstreamer/tensor_query/tensor_query_client.h File Reference

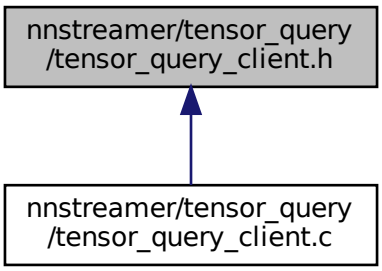
GStreamer plugin to handle tensor query client.

```
#include <gst/gst.h>
#include <gio/gio.h>
#include <tensor_common.h>
#include <nnstreamer-edge.h>
```

Include dependency graph for tensor_query_client.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct _GstTensorQueryClient

  GstTensorQueryClient data structure.
- struct _GstTensorQueryClientClass

  GstTensorQueryClientClass data structure.

Macros

- #define GST_TYPE_TENSOR_QUERY_CLIENT (gst_tensor_query_client_get_type())
- #define GST_TENSOR_QUERY_CLIENT(obj) (G_TYPE_CHECK_INSTANCE_CAST((obj),GST_TYPE_TENSOR_QUERY_C
- #define GST_TENSOR_QUERY_CLIENT_CLASS(klass) (G_TYPE_CHECK_CLASS_CAST((klass),GST_TYPE_TENSOR_Q
- #define GST_IS_TENSOR_QUERY_CLIENT(obj) (G_TYPE_CHECK_INSTANCE_TYPE((obj),GST_TYPE_TENSOR_QUERY
- #define GST_IS_TENSOR_QUERY_CLIENT_CLASS(klass) (G_TYPE_CHECK_CLASS_TYPE((klass),GST_TYPE_TENSOR
- #define GST_TENSOR_QUERY_CLIENT_CAST(obj) ((GstTensorQueryClient *)(obj))

Typedefs

- typedef struct _GstTensorQueryClient GstTensorQueryClient
- typedef struct _GstTensorQueryClientClass GstTensorQueryClientClass

Functions

- GType gst_tensor_query_client_get_type (void)

9.116.1 Detailed Description

GStreamer plugin to handle tensor query client.

Date

09 Jul 2021

Author

Junhwan Kim jejudo.kim@samsung.com

See also

http://github.com/nnstreamer/nnstreamer

Bug No known bugs

9.116.2 Macro Definition Documentation

9.116.2.1 GST_IS_TENSOR_QUERY_CLIENT

```
#define GST_IS_TENSOR_QUERY_CLIENT(
              obj ) (G_TYPE_CHECK_INSTANCE_TYPE((obj),GST_TYPE_TENSOR_QUERY_CLIENT))
```

Definition at line 29 of file tensor_query_client.h.

9.116.2.2 GST_IS_TENSOR_QUERY_CLIENT_CLASS

```
#define GST_IS_TENSOR_QUERY_CLIENT_CLASS(
              klass ) (G_TYPE_CHECK_CLASS_TYPE((klass),GST_TYPE_TENSOR_QUERY_CLIENT))
```

Definition at line 31 of file tensor_query_client.h.

9.116.2.3 GST_TENSOR_QUERY_CLIENT

```
#define GST_TENSOR_QUERY_CLIENT(
              obj ) (G_TYPE_CHECK_INSTANCE_CAST((obj),GST_TYPE_TENSOR_QUERY_CLIENT,GstTensorQueryClient))
```

Definition at line 25 of file tensor_query_client.h.

9.116.2.4 GST_TENSOR_QUERY_CLIENT_CAST

```
#define GST_TENSOR_QUERY_CLIENT_CAST(
              obj ) ((GstTensorQueryClient *)(obj))
```

Definition at line 33 of file tensor_query_client.h.

9.116.2.5 GST_TENSOR_QUERY_CLIENT_CLASS

```
#define GST_TENSOR_QUERY_CLIENT_CLASS(
              klass ) (G_TYPE_CHECK_CLASS_CAST((klass),GST_TYPE_TENSOR_QUERY_CLIENT,GstTensorQueryClientClass)
```

Definition at line 27 of file tensor_query_client.h.

9.116.2.6 GST_TYPE_TENSOR_QUERY_CLIENT

```
#define GST_TYPE_TENSOR_QUERY_CLIENT (gst_tensor_query_client_get_type())
```

Definition at line 23 of file tensor_query_client.h.

9.116.3 Typedef Documentation

9.116.3.1 GstTensorQueryClient

```
typedef struct _GstTensorQueryClient GstTensorQueryClient
```

Definition at line 35 of file tensor_query_client.h.

9.116.3.2 GstTensorQueryClientClass

```
typedef struct _GstTensorQueryClientClass GstTensorQueryClientClass
```

Definition at line 36 of file tensor_query_client.h.

9.116.4 Function Documentation

9.116.4.1 gst_tensor_query_client_get_type()

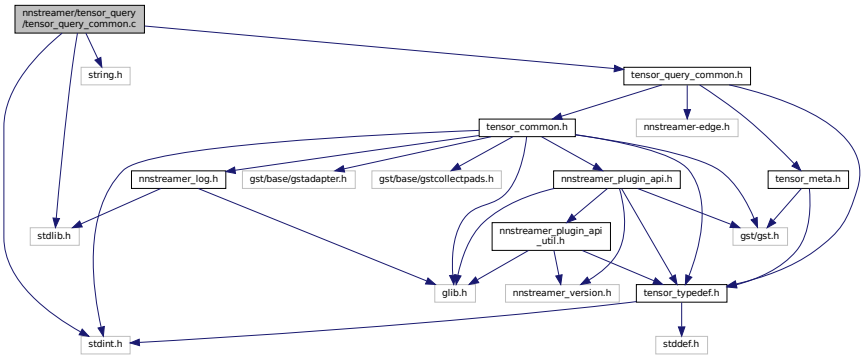
```
GType gst_tensor_query_client_get_type (
            void  )
```

9.117 nnstreamer/tensor_query/tensor_query_common.c File Reference

Utility functions for tensor query.

```
#include <stdint.h>
#include <stdlib.h>
#include <string.h>
#include "tensor_query_common.h"
```

Include dependency graph for tensor_query_common.c:



Macros

- #define EREMOTEIO 121 /∗ This is Linux-specific. Define this for non-Linux systems ∗/

Functions

- GType gst_tensor_query_get_connect_type (void)

  Register GEnumValue array for query connect-type property.

9.117.1 Detailed Description

Utility functions for tensor query.

Copyright (C) 2021 Gichan Jang gichan2.jang@samsung.com

Date

09 July 2021

See also

https://github.com/nnstreamer/nnstreamer

Author

Gichan Jang gichan2.jang@samsung.com

Junhwan Kim jejudo.kim@samsung.com

Bug No known bugs except for NYI items

9.117.2 Macro Definition Documentation

9.117.2.1 EREMOTEIO

```
#define EREMOTEIO 121 /* This is Linux-specific.  Define this for non-Linux systems */
```

Definition at line 23 of file tensor_query_common.c.

9.117.3 Function Documentation

9.117.3.1 gst_tensor_query_get_connect_type()

```
GType gst_tensor_query_get_connect_type (
            void  )
```

Register GEnumValue array for query connect-type property.

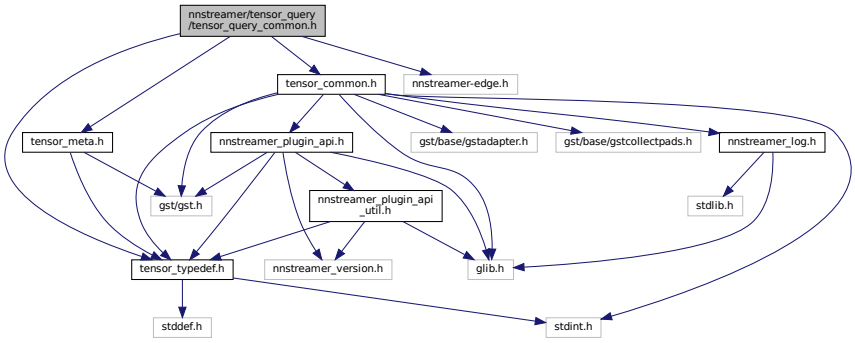
Definition at line 30 of file tensor_query_common.c.

9.118 nnstreamer/tensor_query/tensor_query_common.h File Reference

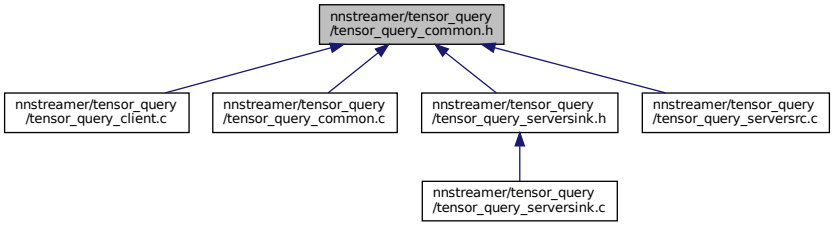
Utility functions for tensor query.

```
#include "tensor_typedef.h"
#include "tensor_common.h"
#include "tensor_meta.h"
#include <nnstreamer-edge.h>
```

Include dependency graph for tensor_query_common.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define QUERY_DEFAULT_TIMEOUT_SEC 10

  Default timeout, in seconds.
- #define DEFAULT_HOST "localhost"

  protocol options for tensor query.
- #define DEFAULT_CONNECT_TYPE (NNS_EDGE_CONNECT_TYPE_TCP)
- #define GST_TYPE_QUERY_CONNECT_TYPE (gst_tensor_query_get_connect_type ())

Functions

- GType gst_tensor_query_get_connect_type (void)

  Register GEnumValue array for query connect-type property.

9.118.1 Detailed Description

Utility functions for tensor query.

Copyright (C) 2021 Gichan Jang gichan2.jang@samsung.com

Date

09 July 2021

See also

https://github.com/nnstreamer/nnstreamer

Author

Gichan Jang gichan2.jang@samsung.com

Bug No known bugs except for NYI items

9.118.2 Macro Definition Documentation

9.118.2.1 DEFAULT_CONNECT_TYPE

```
#define DEFAULT_CONNECT_TYPE (NNS_EDGE_CONNECT_TYPE_TCP)
```

Definition at line 34 of file tensor_query_common.h.

9.118.2.2 DEFAULT_HOST

```
#define DEFAULT_HOST "localhost"
```

protocol options for tensor query.

Definition at line 33 of file tensor_query_common.h.

9.118.2.3 GST_TYPE_QUERY_CONNECT_TYPE

```
#define GST_TYPE_QUERY_CONNECT_TYPE (gst_tensor_query_get_connect_type ())
```

Definition at line 35 of file tensor_query_common.h.

9.118.2.4 QUERY_DEFAULT_TIMEOUT_SEC

```
#define QUERY_DEFAULT_TIMEOUT_SEC 10
```

Default timeout, in seconds.

Definition at line 28 of file tensor_query_common.h.

9.118.3 Function Documentation

9.118.3.1 gst_tensor_query_get_connect_type()

```
GType gst_tensor_query_get_connect_type (
            void  )
```

Register GEnumValue array for query connect-type property.

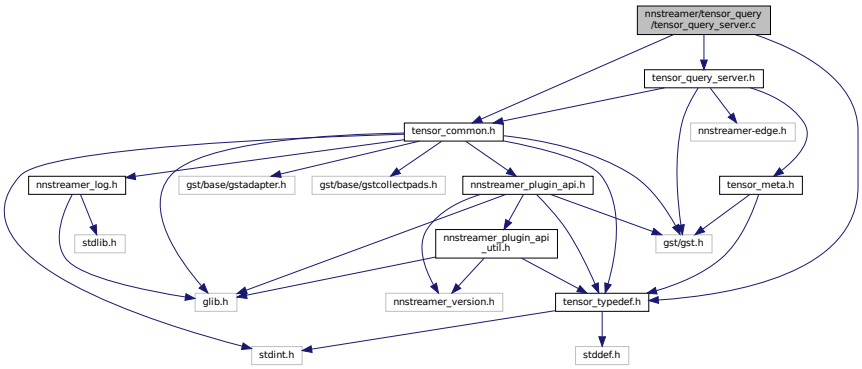
Definition at line 30 of file tensor_query_common.c.

9.119 nnstreamer/tensor_query/tensor_query_server.c File Reference

GStreamer plugin to handle meta_query for server elements.

```
#include "tensor_query_server.h"
#include <tensor_typedef.h>
#include <tensor_common.h>
```

Include dependency graph for tensor_query_server.c:



Functions

- G_LOCK_DEFINE_STATIC (query_server_table)

  mutex for tensor-query server table.
- static void init_queryserver (void)

  Internal function to release query server data.
- static GstTensorQueryServer * gst_tensor_query_server_get_handle (const guint id)

  Get nnstreamer edge server handle.
- gboolean gst_tensor_query_server_add_data (const guint id)

  Add nnstreamer edge server handle into hash table.
- gboolean gst_tensor_query_server_prepare (const guint id, nns_edge_connect_type_e connect_type, GstTensorQueryEdgeInfo *edge_info)

  Prepare edge connection and its handle.
- gboolean gst_tensor_query_server_send_buffer (const guint id, GstBuffer *buffer)

  Send buffer to connected edge device.
- void gst_tensor_query_server_release_edge_handle (const guint id)

  Release nnstreamer edge handle of query server.
- void gst_tensor_query_server_remove_data (const guint id)

  Remove GstTensorQueryServer.
- gboolean gst_tensor_query_server_wait_sink (const guint id)

  Wait until the sink is configured and get server info handle.
- void gst_tensor_query_server_set_configured (const guint id)

  set query server sink configured.
- void gst_tensor_query_server_set_caps (const guint id, const gchar *caps_str)

  set query server caps.
- static void fini_queryserver (void)

  Destruct the query server.

Variables

- static GHashTable * _qs_table = NULL

  Table for query server data.

9.119.1 Detailed Description

GStreamer plugin to handle meta_query for server elements.

Date

03 Aug 2021

Author

Junhwan Kim jejudo.kim@samsung.com

See also

http://github.com/nnstreamer/nnstreamer

Bug No known bugs

9.119.2 Function Documentation

9.119.2.1 fini_queryserver()

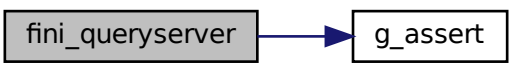
```
static void fini_queryserver (
            void  )  [static]
```

Destruct the query server.

Internal error (init not called?)

Definition at line 385 of file tensor_query_server.c.

Here is the call graph for this function:



9.119.2.2 G_LOCK_DEFINE_STATIC()

```
G_LOCK_DEFINE_STATIC (
            query_server_table  )
```

mutex for tensor-query server table.

9.119.2.3 gst_tensor_query_server_add_data()

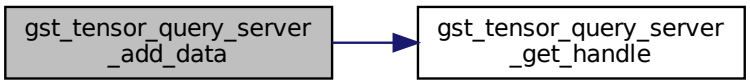
```
gboolean gst_tensor_query_server_add_data (
            const guint id )
```

Add nnstreamer edge server handle into hash table.

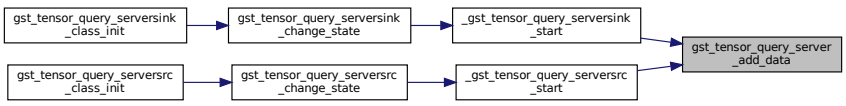
Add GstTensorQueryServer.

Definition at line 77 of file tensor_query_server.c.

Here is the call graph for this function:



Here is the caller graph for this function:



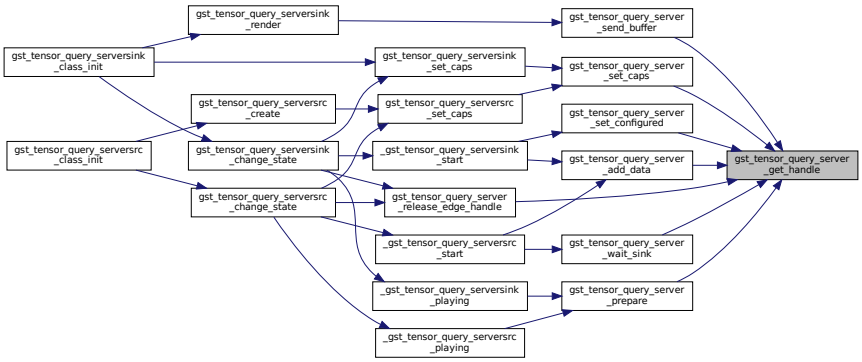
9.119.2.4 gst_tensor_query_server_get_handle()

```
static GstTensorQueryServer* gst_tensor_query_server_get_handle (
            const guint id )  [static]
```

Get nnstreamer edge server handle.

Definition at line 62 of file tensor_query_server.c.

Here is the caller graph for this function:



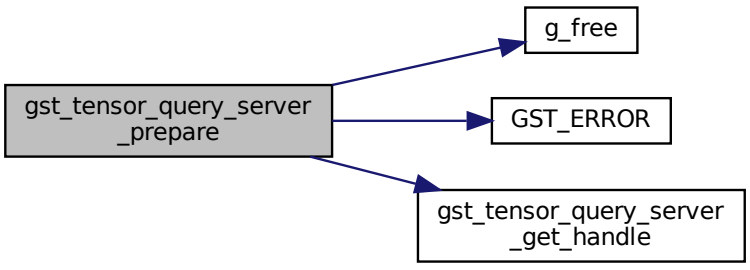
9.119.2.5 gst_tensor_query_server_prepare()

```
gboolean gst_tensor_query_server_prepare (
            const guint id,
            nns_edge_connect_type_e connect_type,
            GstTensorQueryEdgeInfo * edge_info )
```

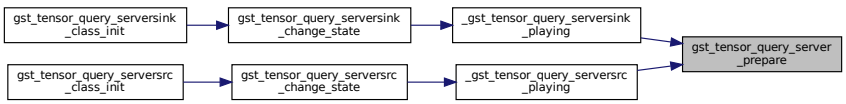
Prepare edge connection and its handle.

Definition at line 114 of file tensor_query_server.c.

Here is the call graph for this function:



Here is the caller graph for this function:



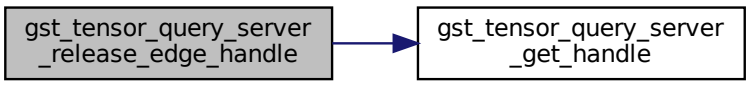
9.119.2.6 gst_tensor_query_server_release_edge_handle()

```
void gst_tensor_query_server_release_edge_handle (
             const guint id )
```

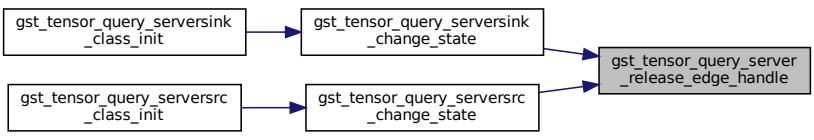
Release nnstreamer edge handle of query server.

Definition at line 258 of file tensor_query_server.c.

Here is the call graph for this function:



Here is the caller graph for this function:



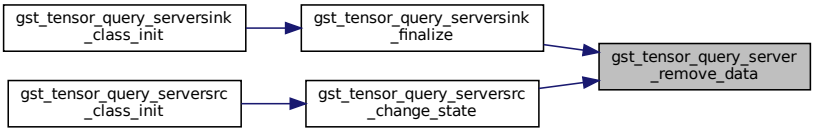
9.119.2.7 gst_tensor_query_server_remove_data()

```
void gst_tensor_query_server_remove_data (
            const guint id )
```

Remove GstTensorQueryServer.

Definition at line 280 of file tensor_query_server.c.

Here is the caller graph for this function:



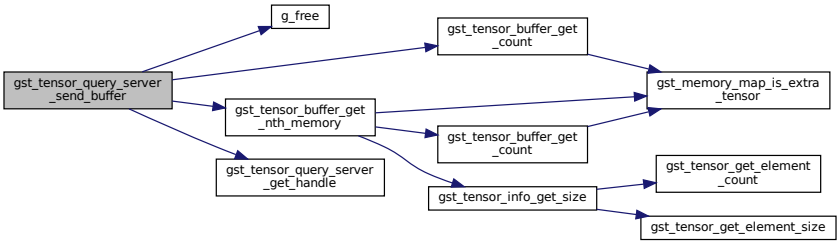
9.119.2.8 gst_tensor_query_server_send_buffer()

```
gboolean gst_tensor_query_server_send_buffer (
            const guint id,
            GstBuffer * buffer )
```

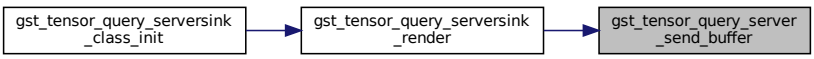
Send buffer to connected edge device.

Definition at line 183 of file tensor_query_server.c.

Here is the call graph for this function:



Here is the caller graph for this function:



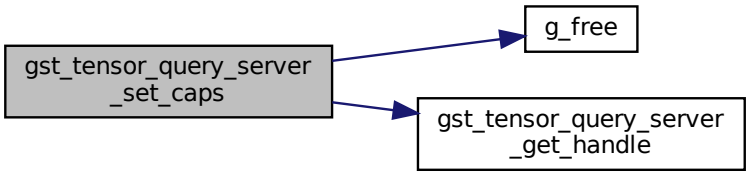
9.119.2.9 gst_tensor_query_server_set_caps()

```
void gst_tensor_query_server_set_caps (
            const guint id,
            const gchar * caps_str )
```

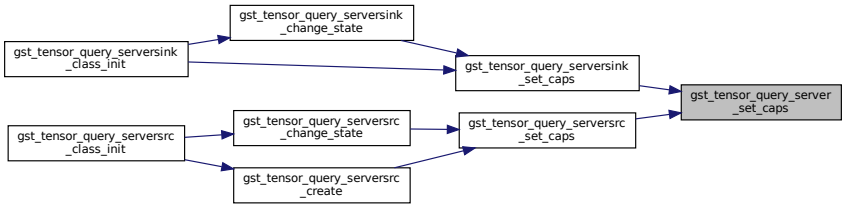
set query server caps.

Definition at line 342 of file tensor_query_server.c.

Here is the call graph for this function:



Here is the caller graph for this function:



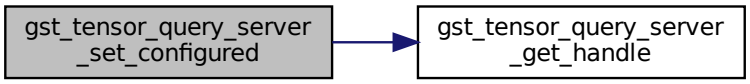
9.119.2.10 gst_tensor_query_server_set_configured()

```
void gst_tensor_query_server_set_configured (
            const guint id )
```

set query server sink configured.

Definition at line 322 of file tensor_query_server.c.

Here is the call graph for this function:



Here is the caller graph for this function:

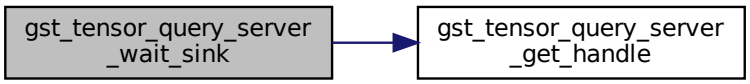
9.119.2.11 gst_tensor_query_server_wait_sink()

```
gboolean gst_tensor_query_server_wait_sink (
            const guint id )
```

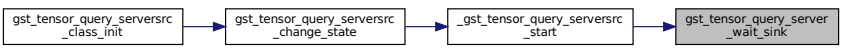
Wait until the sink is configured and get server info handle.

Definition at line 292 of file tensor_query_server.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.119.2.12 init_queryserver()

```
static void init_queryserver (
            void  )  [static]
```

Internal function to release query server data.

Initialize the query server. Internal error (duplicated init call?)

Definition at line 31 of file tensor_query_server.c.

9.119.3 Variable Documentation

9.119.3.1 _qs_table

```
GHashTable* _qs_table = NULL  [static]
```

Table for query server data.

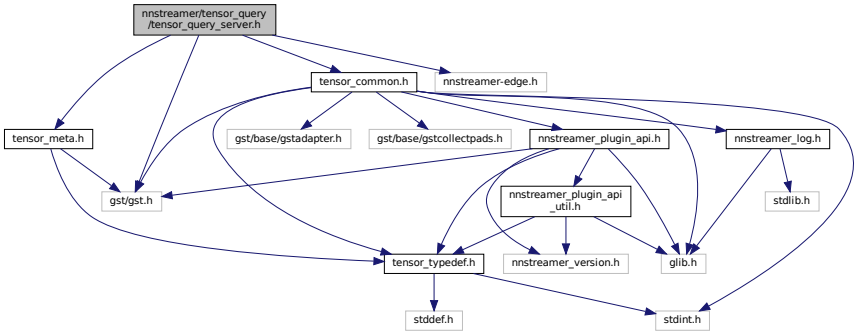
Definition at line 29 of file tensor_query_server.c.

9.120 nnstreamer/tensor_query/tensor_query_server.h File Reference

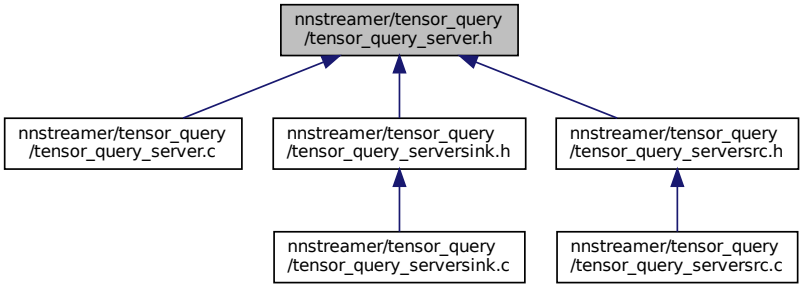
GStreamer plugin to handle meta_query for server elements.

```
#include <gst/gst.h>
#include <tensor_common.h>
#include <nnstreamer-edge.h>
#include "tensor_meta.h"
```

Include dependency graph for tensor_query_server.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct GstTensorQueryEdgeInfo

  Internal data structure for nns-edge info to prepare edge connection.
- struct GstTensorQueryServer

  GstTensorQueryServer internal info data structure.

Macros

- #define DEFAULT_SERVER_ID 0
- #define DEFAULT_QUERY_INFO_TIMEOUT 5

Functions

- gboolean gst_tensor_query_server_add_data (const guint id)

  Add GstTensorQueryServer.
- gboolean gst_tensor_query_server_prepare (const guint id, nns_edge_connect_type_e connect_type, GstTensorQueryEdgeInfo ∗edge_info)

  Prepare edge connection and its handle.
- void gst_tensor_query_server_remove_data (const guint id)

  Remove GstTensorQueryServer.
- gboolean gst_tensor_query_server_wait_sink (const guint id)

  Wait until the sink is configured and get server info handle.
- gboolean gst_tensor_query_server_send_buffer (const guint id, GstBuffer ∗buffer)

  Send buffer to connected edge device.
- void gst_tensor_query_server_set_configured (const guint id)

  set query server sink configured.
- void gst_tensor_query_server_set_caps (const guint id, const gchar ∗caps_str)

  set query server caps.
- void gst_tensor_query_server_release_edge_handle (const guint id)

  Release nnstreamer edge handle of query server.

9.120.1 Detailed Description

GStreamer plugin to handle meta_query for server elements.

Date

03 Aug 2021

Author

Junhwan Kim jejudo.kim@samsung.com

See also

http://github.com/nnstreamer/nnstreamer

Bug No known bugs

9.120.2 Macro Definition Documentation

9.120.2.1 DEFAULT_QUERY_INFO_TIMEOUT

```
#define DEFAULT_QUERY_INFO_TIMEOUT 5
```

Definition at line 23 of file tensor_query_server.h.

9.120.2.2 DEFAULT_SERVER_ID

```
#define DEFAULT_SERVER_ID 0
```

Definition at line 22 of file tensor_query_server.h.

9.120.3 Function Documentation

9.120.3.1 gst_tensor_query_server_add_data()

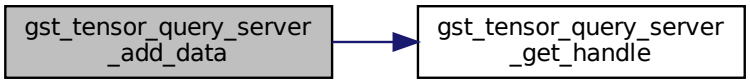
```
gboolean gst_tensor_query_server_add_data (
            const guint id )
```

Add GstTensorQueryServer.

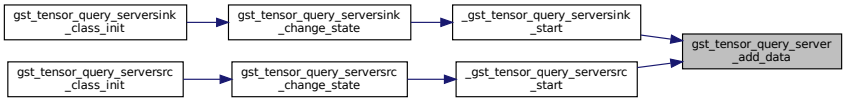
Add GstTensorQueryServer.

Definition at line 77 of file tensor_query_server.c.

Here is the call graph for this function:



Here is the caller graph for this function:



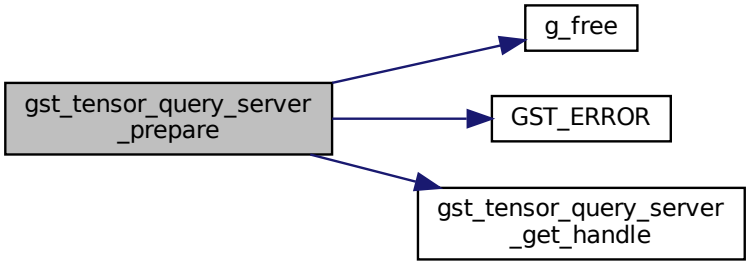
9.120.3.2 gst_tensor_query_server_prepare()

```
gboolean gst_tensor_query_server_prepare (
            const guint id,
            nns_edge_connect_type_e connect_type,
            GstTensorQueryEdgeInfo * edge_info )
```

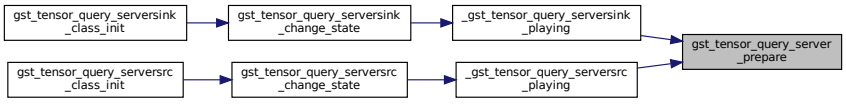
Prepare edge connection and its handle.

Definition at line 114 of file tensor_query_server.c.

Here is the call graph for this function:



Here is the caller graph for this function:



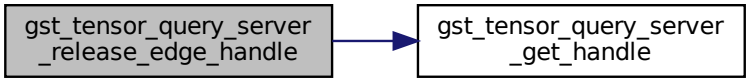
9.120.3.3 gst_tensor_query_server_release_edge_handle()

```
void gst_tensor_query_server_release_edge_handle (
             const guint id )
```

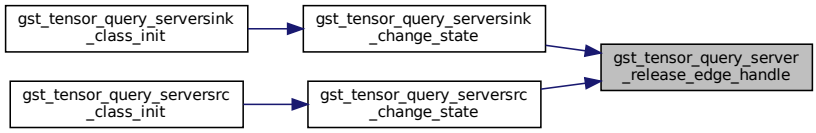
Release nnstreamer edge handle of query server.

Definition at line 258 of file tensor_query_server.c.

Here is the call graph for this function:



Here is the caller graph for this function:



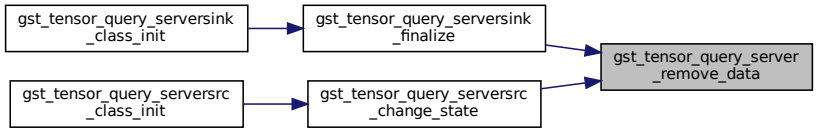
9.120.3.4 gst_tensor_query_server_remove_data()

```
void gst_tensor_query_server_remove_data (
            const guint id )
```

Remove GstTensorQueryServer.

Definition at line 280 of file tensor_query_server.c.

Here is the caller graph for this function:



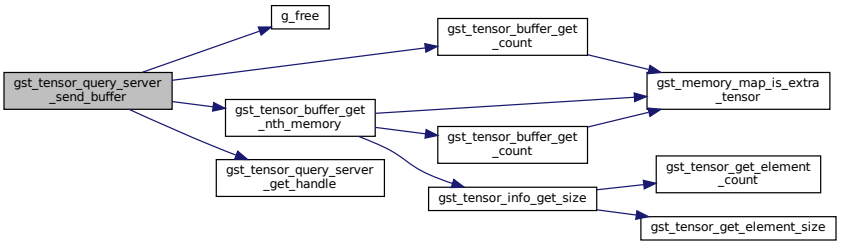
9.120.3.5 gst_tensor_query_server_send_buffer()

```
gboolean gst_tensor_query_server_send_buffer (
            const guint id,
            GstBuffer * buffer )
```

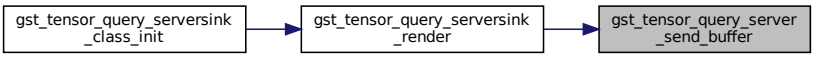
Send buffer to connected edge device.

Definition at line 183 of file tensor_query_server.c.

Here is the call graph for this function:



Here is the caller graph for this function:



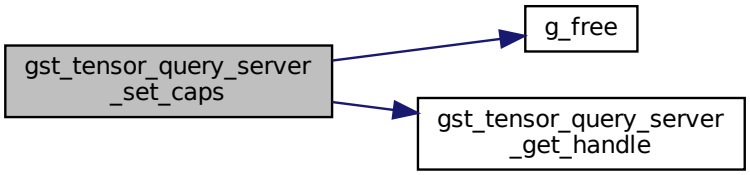
9.120.3.6 gst_tensor_query_server_set_caps()

```
void gst_tensor_query_server_set_caps (
            const guint id,
            const gchar * caps_str )
```

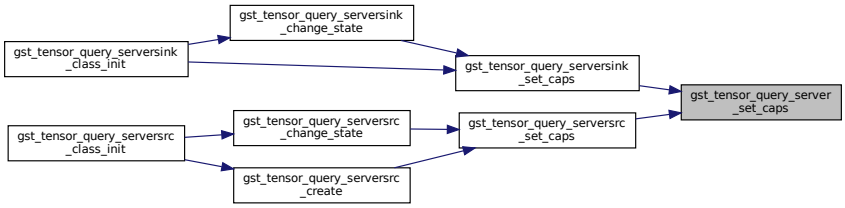
set query server caps.

Definition at line 342 of file tensor_query_server.c.

Here is the call graph for this function:



Here is the caller graph for this function:



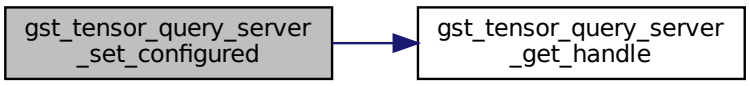
9.120.3.7 gst_tensor_query_server_set_configured()

```
void gst_tensor_query_server_set_configured (
             const guint id )
```

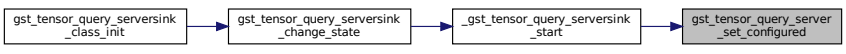
set query server sink configured.

Definition at line 322 of file tensor_query_server.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.120.3.8 gst_tensor_query_server_wait_sink()

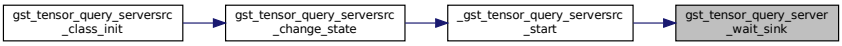
```
gboolean gst_tensor_query_server_wait_sink (
            const guint id )
```

Wait until the sink is configured and get server info handle.

Definition at line 292 of file tensor_query_server.c.

Here is the call graph for this function:

Here is the caller graph for this function:

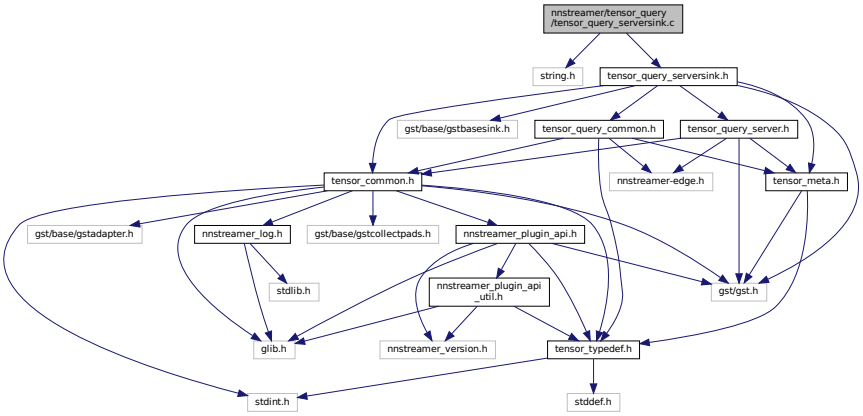


9.121 nnstreamer/tensor_query/tensor_query_serversink.c File Reference

GStreamer plugin to handle tensor query server sink.

```
#include <string.h>
#include "tensor_query_serversink.h"
```

Include dependency graph for tensor_query_serversink.c:



Macros

- #define GST_CAT_DEFAULT gst_tensor_query_serversink_debug
- #define DEFAULT_METALESS_FRAME_LIMIT 1
- #define gst_tensor_query_serversink_parent_class parent_class

Enumerations

- enum {
  PROP_0, PROP_CONNECT_TYPE, PROP_ID, PROP_TIMEOUT,
  PROP_METALESS_FRAME_LIMIT }

Functions

- GST_DEBUG_CATEGORY_STATIC (gst_tensor_query_serversink_debug)
- G_DEFINE_TYPE (GstTensorQueryServerSink, gst_tensor_query_serversink, GST_TYPE_BASE_SINK)
- static GstStateChangeReturn gst_tensor_query_serversink_change_state (GstElement *element, GstStateChange transition)

  Change state of query server sink.
- static void gst_tensor_query_serversink_set_property (GObject *object, guint prop_id, const GValue *value, GParamSpec *pspec)

  set property
- static void gst_tensor_query_serversink_get_property (GObject *object, guint prop_id, GValue *value, GParamSpec *pspec)

  get property
- static void gst_tensor_query_serversink_finalize (GObject *object)

  finalize the object
- static GstFlowReturn gst_tensor_query_serversink_render (GstBaseSink *bsink, GstBuffer *buf)

  render buffer, send buffer to client
- static gboolean gst_tensor_query_serversink_set_caps (GstBaseSink *bsink, GstCaps *caps)

  An implementation of the set_caps vmethod in GstBaseSinkClass.
- static void gst_tensor_query_serversink_class_init (GstTensorQueryServerSinkClass *klass)

  initialize the class
- static void gst_tensor_query_serversink_init (GstTensorQueryServerSink *sink)

  initialize the new element
- static gboolean _gst_tensor_query_serversink_start (GstTensorQueryServerSink *sink)

  start processing of query_serversink
- static gboolean _gst_tensor_query_serversink_playing (GstTensorQueryServerSink *sink)

  start processing of query_serversink

Variables

- static GstStaticPadTemplate sinktemplate

  the capabilities of the inputs.

9.121.1 Detailed Description

GStreamer plugin to handle tensor query server sink.

Date

09 Jul 2021

Author

Junhwan Kim jejudo.kim@samsung.com

See also

http://github.com/nnstreamer/nnstreamer

Bug No known bugs

9.121.2 Macro Definition Documentation

9.121.2.1 DEFAULT_METALESS_FRAME_LIMIT

```
#define DEFAULT_METALESS_FRAME_LIMIT 1
```

Definition at line 23 of file tensor_query_serversink.c.

9.121.2.2 GST_CAT_DEFAULT

```
#define GST_CAT_DEFAULT gst_tensor_query_serversink_debug
```

Definition at line 21 of file tensor_query_serversink.c.

9.121.2.3 gst_tensor_query_serversink_parent_class

```
#define gst_tensor_query_serversink_parent_class parent_class
```

Definition at line 42 of file tensor_query_serversink.c.

9.121.3 Enumeration Type Documentation

9.121.3.1 anonymous enum

```
anonymous enum
```

Enumerator

PROP_0	
PROP_CONNECT_TYPE	
PROP_ID	
PROP_TIMEOUT	
PROP_METALESS_FRAME_LIMIT	

Definition at line 33 of file tensor_query_serversink.c.

9.121.4 Function Documentation

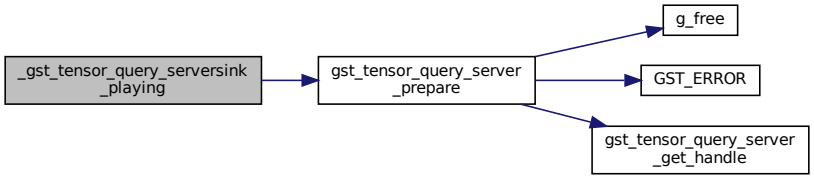
9.121.4.1 _gst_tensor_query_serversink_playing()

```
static gboolean _gst_tensor_query_serversink_playing (
            GstTensorQueryServerSink * sink )  [static]
```

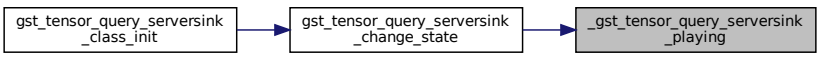
start processing of query_serversink

Definition at line 156 of file tensor_query_serversink.c.

Here is the call graph for this function:



Here is the caller graph for this function:



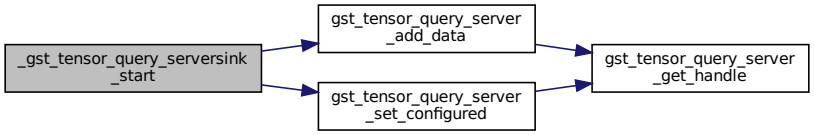
9.121.4.2 _gst_tensor_query_serversink_start()

```
static gboolean _gst_tensor_query_serversink_start (
            GstTensorQueryServerSink * sink )  [static]
```

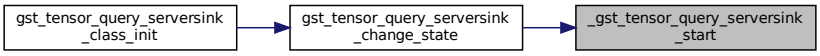
start processing of query_serversink

Definition at line 141 of file tensor_query_serversink.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.121.4.3 G_DEFINE_TYPE()

```
G_DEFINE_TYPE (
            GstTensorQueryServerSink ,
            gst_tensor_query_serversink ,
            GST_TYPE_BASE_SINK  )
```

9.121.4.4 GST_DEBUG_CATEGORY_STATIC()

```
GST_DEBUG_CATEGORY_STATIC (
            gst_tensor_query_serversink_debug  )
```

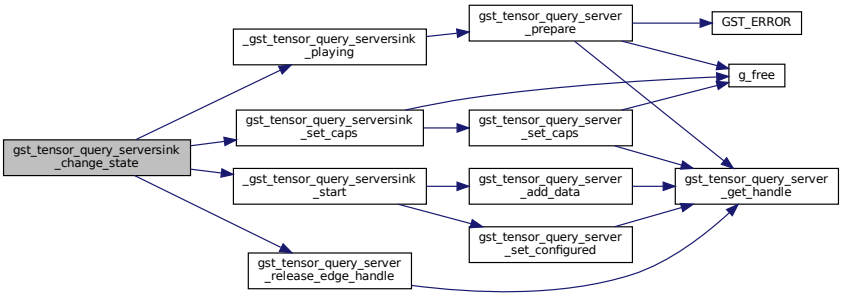
9.121.4.5 gst_tensor_query_serversink_change_state()

```
static GstStateChangeReturn gst_tensor_query_serversink_change_state (
            GstElement * element,
            GstStateChange transition )  [static]
```

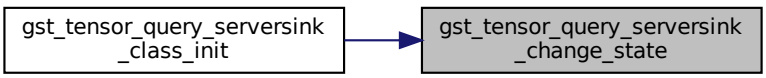
Change state of query server sink.

Definition at line 170 of file tensor_query_serversink.c.

Here is the call graph for this function:



Here is the caller graph for this function:



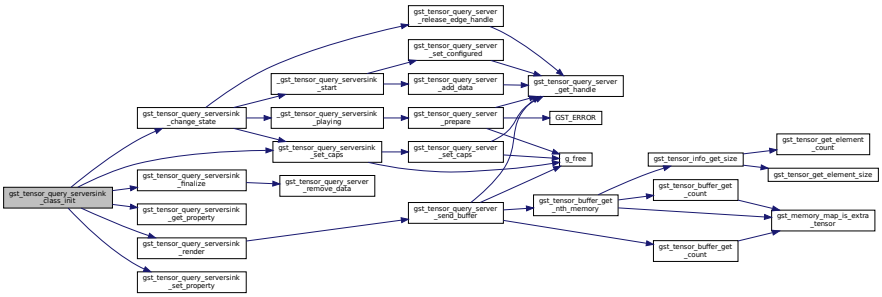
9.121.4.6 gst_tensor_query_serversink_class_init()

```
static void gst_tensor_query_serversink_class_init (
            GstTensorQueryServerSinkClass * klass )  [static]
```

initialize the class

Definition at line 62 of file tensor_query_serversink.c.

Here is the call graph for this function:



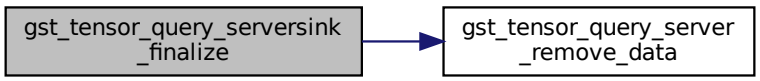
9.121.4.7 gst_tensor_query_serversink_finalize()

```
static void gst_tensor_query_serversink_finalize (
            GObject * object )  [static]
```

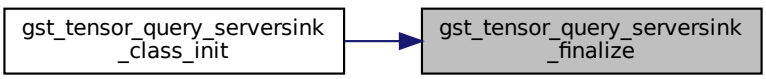
finalize the object

Definition at line 130 of file tensor_query_serversink.c.

Here is the call graph for this function:



Here is the caller graph for this function:



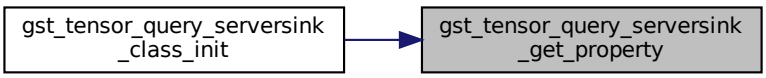
9.121.4.8 gst_tensor_query_serversink_get_property()

```
static void gst_tensor_query_serversink_get_property (
            GObject * object,
            guint prop_id,
            GValue * value,
            GParamSpec * pspec )  [static]
```

get property

Definition at line 248 of file tensor_query_serversink.c.

Here is the caller graph for this function:



9.121.4.9 gst_tensor_query_serversink_init()

```
static void gst_tensor_query_serversink_init (
            GstTensorQueryServerSink * sink )  [static]
```

initialize the new element

Definition at line 118 of file tensor_query_serversink.c.

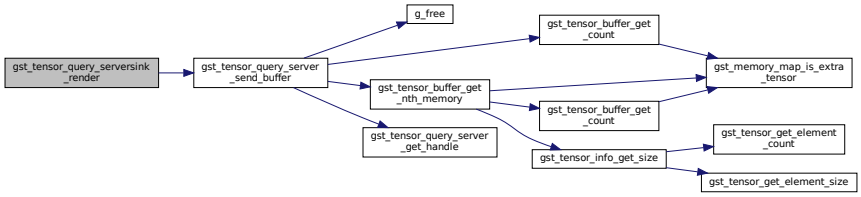
9.121.4.10 gst_tensor_query_serversink_render()

```
static GstFlowReturn gst_tensor_query_serversink_render (
            GstBaseSink * bsink,
            GstBuffer * buf )  [static]
```

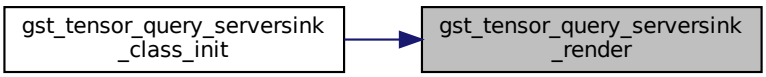
render buffer, send buffer to client

Definition at line 296 of file tensor_query_serversink.c.

Here is the call graph for this function:



Here is the caller graph for this function:



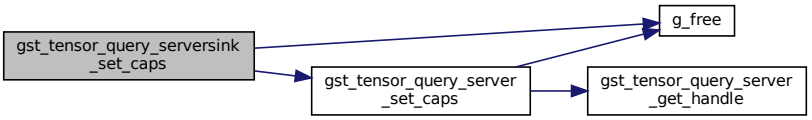
9.121.4.11 gst_tensor_query_serversink_set_caps()

```
static gboolean gst_tensor_query_serversink_set_caps (
            GstBaseSink * basesink,
            GstCaps * caps )  [static]
```

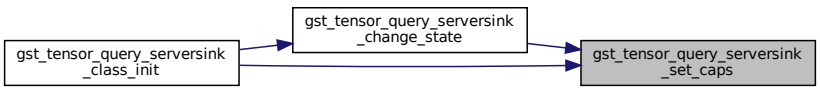
An implementation of the set_caps vmethod in GstBaseSinkClass.

Definition at line 276 of file tensor_query_serversink.c.

Here is the call graph for this function:



Here is the caller graph for this function:



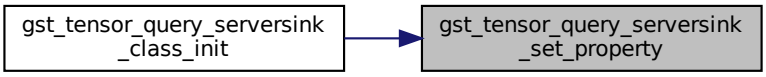
9.121.4.12 gst_tensor_query_serversink_set_property()

```
static void gst_tensor_query_serversink_set_property (
             GObject * object,
             guint prop_id,
             const GValue * value,
             GParamSpec * pspec )  [static]
```

set property

Definition at line 220 of file tensor_query_serversink.c.

Here is the caller graph for this function:



9.121.5 Variable Documentation

9.121.5.1 sinktemplate

```
GstStaticPadTemplate sinktemplate  [static]
```

Initial value:

```
= GST_STATIC_PAD_TEMPLATE ("sink",
    GST_PAD_SINK,
    GST_PAD_ALWAYS,
    GST_STATIC_CAPS_ANY)
```

the capabilities of the inputs.

Definition at line 28 of file tensor_query_serversink.c.

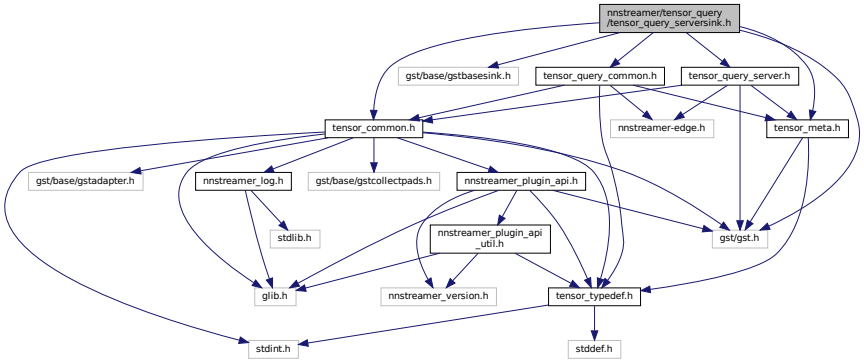
9.122 nnstreamer/tensor_query/tensor_query_serversink.h File Reference

GStreamer plugin to handle tensor query_server sink.

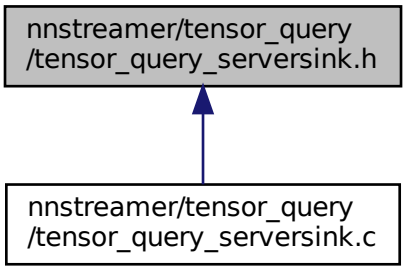
```
#include <gst/gst.h>
#include <gst/base/gstbasesink.h>
#include <tensor_common.h>
#include <tensor_meta.h>
#include "tensor_query_server.h"
```

```
#include "tensor_query_common.h"
```

Include dependency graph for tensor_query_serversink.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct _GstTensorQueryServerSink

  GstTensorQueryServerSink data structure.
- struct _GstTensorQueryServerSinkClass

  GstTensorQueryServerSinkClass data structure.

Macros

- #define GST_TYPE_TENSOR_QUERY_SERVERSINK (gst_tensor_query_serversink_get_type())
- #define GST_TENSOR_QUERY_SERVERSINK(obj) (G_TYPE_CHECK_INSTANCE_CAST((obj),GST_TYPE_TENSOR_QUE
- #define GST_TENSOR_QUERY_SERVERSINK_CLASS(klass) (G_TYPE_CHECK_CLASS_CAST((klass),GST_TYPE_TENS
- #define GST_IS_TENSOR_QUERY_SERVERSINK(obj) (G_TYPE_CHECK_INSTANCE_TYPE((obj),GST_TYPE_TENSOR_C
- #define GST_IS_TENSOR_QUERY_SERVERSINK_CLASS(klass) (G_TYPE_CHECK_CLASS_TY↩ PE((klass),GST_TYPE_TENSOR_QUERY_SERVERSINK))
- #define GST_TENSOR_QUERY_SERVERSINK_CAST(obj) ((GstTensorQueryServerSink *)(obj))

Typedefs

- typedef struct _GstTensorQueryServerSink GstTensorQueryServerSink
- typedef struct _GstTensorQueryServerSinkClass GstTensorQueryServerSinkClass

Functions

- GType gst_tensor_query_serversink_get_type (void)

9.122.1 Detailed Description

GStreamer plugin to handle tensor query_server sink.

Date

09 Jul 2021

Author

Junhwan Kim jejudo.kim@samsung.com

See also

http://github.com/nnstreamer/nnstreamer

Bug No known bugs

9.122.2 Macro Definition Documentation

9.122.2.1 GST_IS_TENSOR_QUERY_SERVERSINK

```
#define GST_IS_TENSOR_QUERY_SERVERSINK(
              obj ) (G_TYPE_CHECK_INSTANCE_TYPE((obj),GST_TYPE_TENSOR_QUERY_SERVERSINK))
```

Definition at line 29 of file tensor_query_serversink.h.

9.122.2.2 GST_IS_TENSOR_QUERY_SERVERSINK_CLASS

```
#define GST_IS_TENSOR_QUERY_SERVERSINK_CLASS(
              klass ) (G_TYPE_CHECK_CLASS_TYPE((klass),GST_TYPE_TENSOR_QUERY_SERVERSINK))
```

Definition at line 31 of file tensor_query_serversink.h.

9.122.2.3 GST_TENSOR_QUERY_SERVERSINK

```
#define GST_TENSOR_QUERY_SERVERSINK(
              obj ) (G_TYPE_CHECK_INSTANCE_CAST((obj),GST_TYPE_TENSOR_QUERY_SERVERSINK,GstTensorQueryServerSin
```

Definition at line 25 of file tensor_query_serversink.h.

9.122.2.4 GST_TENSOR_QUERY_SERVERSINK_CAST

```
#define GST_TENSOR_QUERY_SERVERSINK_CAST(
              obj ) ((GstTensorQueryServerSink *)(obj))
```

Definition at line 33 of file tensor_query_serversink.h.

9.122.2.5 GST_TENSOR_QUERY_SERVERSINK_CLASS

```
#define GST_TENSOR_QUERY_SERVERSINK_CLASS(
              klass ) (G_TYPE_CHECK_CLASS_CAST((klass),GST_TYPE_TENSOR_QUERY_SERVERSINK,GstTensorQueryServerSi
```

Definition at line 27 of file tensor_query_serversink.h.

9.122.2.6 GST_TYPE_TENSOR_QUERY_SERVERSINK

```
#define GST_TYPE_TENSOR_QUERY_SERVERSINK (gst_tensor_query_serversink_get_type())
```

Definition at line 23 of file tensor_query_serversink.h.

9.122.3 Typedef Documentation

9.122.3.1 GstTensorQueryServerSink

```
typedef struct _GstTensorQueryServerSink GstTensorQueryServerSink
```

Definition at line 34 of file tensor_query_serversink.h.

9.122.3.2 GstTensorQueryServerSinkClass

```
typedef struct _GstTensorQueryServerSinkClass GstTensorQueryServerSinkClass
```

Definition at line 35 of file tensor_query_serversink.h.

9.122.4 Function Documentation

9.122.4.1 gst_tensor_query_serversink_get_type()

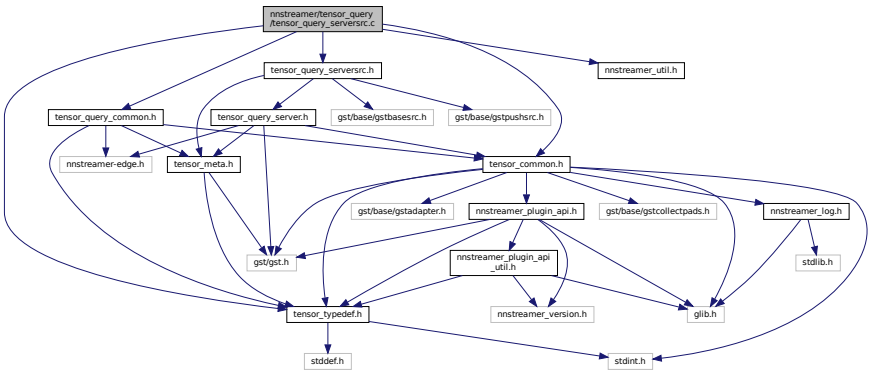
```
GType gst_tensor_query_serversink_get_type (
            void  )
```

9.123 nnstreamer/tensor_query/tensor_query_serversrc.c File Reference

GStreamer plugin to handle tensor query_server src.

```
#include <tensor_typedef.h>
#include <tensor_common.h>
#include "tensor_query_serversrc.h"
#include "tensor_query_common.h"
#include "nnstreamer_util.h"
```

Include dependency graph for tensor_query_serversrc.c:



Macros

- #define GST_CAT_DEFAULT gst_tensor_query_serversrc_debug
- #define DEFAULT_PORT_SRC 3000
- #define DEFAULT_IS_LIVE TRUE
- #define DEFAULT_MQTT_HOST "127.0.0.1"
- #define DEFAULT_MQTT_PORT 1883
- #define DEFAULT_DATA_POP_TIMEOUT 100000U
- #define gst_tensor_query_serversrc_parent_class parent_class

Enumerations

- enum {
  PROP_0, PROP_HOST, PROP_PORT, PROP_DEST_HOST,
  PROP_DEST_PORT, PROP_CONNECT_TYPE, PROP_TIMEOUT, PROP_TOPIC,
  PROP_ID, PROP_IS_LIVE }

  query_serversrc properties

Functions

- GST_DEBUG_CATEGORY_STATIC (gst_tensor_query_serversrc_debug)
- G_DEFINE_TYPE (GstTensorQueryServerSrc, gst_tensor_query_serversrc, GST_TYPE_PUSH_SRC)
- static GstStateChangeReturn gst_tensor_query_serversrc_change_state (GstElement *element, GstState↩ Change transition)

  Change state of query server src.
- static void gst_tensor_query_serversrc_set_property (GObject *object, guint prop_id, const GValue *value, GParamSpec *pspec)

  set property of query_serversrc
- static void gst_tensor_query_serversrc_get_property (GObject *object, guint prop_id, GValue *value, G↩ ParamSpec *pspec)

  get property of query_serversrc
- static void gst_tensor_query_serversrc_finalize (GObject *object)

  finalize the query_serversrc object
- static GstFlowReturn gst_tensor_query_serversrc_create (GstPushSrc *psrc, GstBuffer **outbuf)

  create query_serversrc, wait on socket and receive data
- static gboolean gst_tensor_query_serversrc_set_caps (GstBaseSrc *bsrc, GstCaps *caps)

  An implementation of the set_caps vmethod in GstBaseSrcClass.
- static void gst_tensor_query_serversrc_class_init (GstTensorQueryServerSrcClass *klass)

  initialize the query_serversrc class
- static void gst_tensor_query_serversrc_init (GstTensorQueryServerSrc *src)

  initialize the new query_serversrc element
- static int _nns_edge_event_cb (nns_edge_event_h event_h, void *user_data)

  nnstreamer-edge event callback.
- static gboolean _gst_tensor_query_serversrc_start (GstTensorQueryServerSrc *src)

  start processing of query_serversrc, setting up the server
- static gboolean _gst_tensor_query_serversrc_playing (GstTensorQueryServerSrc *src, nns_edge_↩ connect_type_e connect_type)

  start processing of query_serversrc, setting up the server
- static GstBuffer * _gst_tensor_query_serversrc_get_buffer (GstTensorQueryServerSrc *src)

  Get buffer from message queue.

Variables

- static GstStaticPadTemplate srctemplate

  the capabilities of the outputs

9.123.1 Detailed Description

GStreamer plugin to handle tensor query_server src.

Date

09 Jul 2021

Author

Junhwan Kim jejudo.kim@samsung.com

See also

http://github.com/nnstreamer/nnstreamer

Bug No known bugs

9.123.2 Macro Definition Documentation

9.123.2.1 DEFAULT_DATA_POP_TIMEOUT

```
#define DEFAULT_DATA_POP_TIMEOUT 100000U
```

Definition at line 30 of file tensor_query_serversrc.c.

9.123.2.2 DEFAULT_IS_LIVE

```
#define DEFAULT_IS_LIVE TRUE
```

Definition at line 27 of file tensor_query_serversrc.c.

9.123.2.3 DEFAULT_MQTT_HOST

```
#define DEFAULT_MQTT_HOST "127.0.0.1"
```

Definition at line 28 of file tensor_query_serversrc.c.

9.123.2.4 DEFAULT_MQTT_PORT

```
#define DEFAULT_MQTT_PORT 1883
```

Definition at line 29 of file tensor_query_serversrc.c.

9.123.2.5 DEFAULT_PORT_SRC

```
#define DEFAULT_PORT_SRC 3000
```

Definition at line 26 of file tensor_query_serversrc.c.

9.123.2.6 GST_CAT_DEFAULT

```
#define GST_CAT_DEFAULT gst_tensor_query_serversrc_debug
```

Definition at line 24 of file tensor_query_serversrc.c.

9.123.2.7 gst_tensor_query_serversrc_parent_class

```
#define gst_tensor_query_serversrc_parent_class parent_class
```

Definition at line 57 of file tensor_query_serversrc.c.

9.123.3 Enumeration Type Documentation

9.123.3.1 anonymous enum

```
anonymous enum
```

query_serversrc properties

Enumerator

PROP_0	
PROP_HOST	
PROP_PORT	
PROP_DEST_HOST	
PROP_DEST_PORT	
PROP_CONNECT_TYPE	
PROP_TIMEOUT	
PROP_TOPIC	
PROP_ID	
PROP_IS_LIVE	

Definition at line 43 of file tensor_query_serversrc.c.

9.123.4 Function Documentation

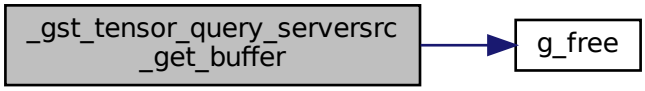
9.123.4.1 _gst_tensor_query_serversrc_get_buffer()

```
static GstBuffer* _gst_tensor_query_serversrc_get_buffer (
            GstTensorQueryServerSrc * src )  [static]
```

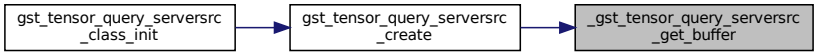
Get buffer from message queue.

Definition at line 434 of file tensor_query_serversrc.c.

Here is the call graph for this function:



Here is the caller graph for this function:



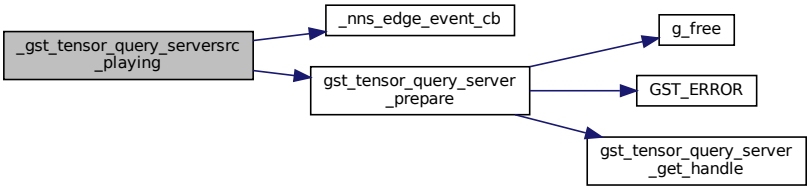
9.123.4.2 _gst_tensor_query_serversrc_playing()

```
static gboolean _gst_tensor_query_serversrc_playing (
            GstTensorQueryServerSrc * src,
            nns_edge_connect_type_e connect_type )  [static]
```

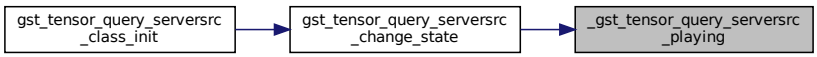
start processing of query_serversrc, setting up the server

Definition at line 252 of file tensor_query_serversrc.c.

Here is the call graph for this function:



Here is the caller graph for this function:



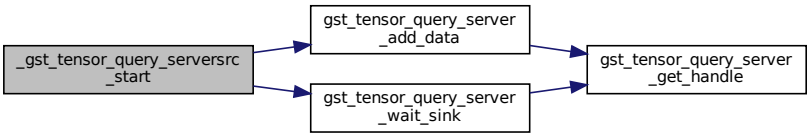
9.123.4.3 _gst_tensor_query_serversrc_start()

```
static gboolean _gst_tensor_query_serversrc_start (
            GstTensorQueryServerSrc * src )  [static]
```

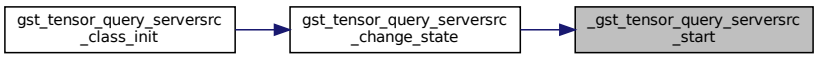
start processing of query_serversrc, setting up the server

Definition at line 235 of file tensor_query_serversrc.c.

Here is the call graph for this function:



Here is the caller graph for this function:



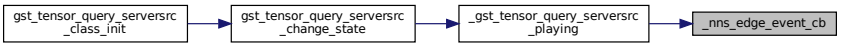
9.123.4.4 _nns_edge_event_cb()

```
static int _nns_edge_event_cb (
            nns_edge_event_h event_h,
            void * user_data )  [static]
```

nnstreamer-edge event callback.

Definition at line 199 of file tensor_query_serversrc.c.

Here is the caller graph for this function:



9.123.4.5 G_DEFINE_TYPE()

```
G_DEFINE_TYPE (
            GstTensorQueryServerSrc ,
            gst_tensor_query_serversrc ,
            GST_TYPE_PUSH_SRC  )
```

9.123.4.6 GST_DEBUG_CATEGORY_STATIC()

```
GST_DEBUG_CATEGORY_STATIC (
            gst_tensor_query_serversrc_debug  )
```

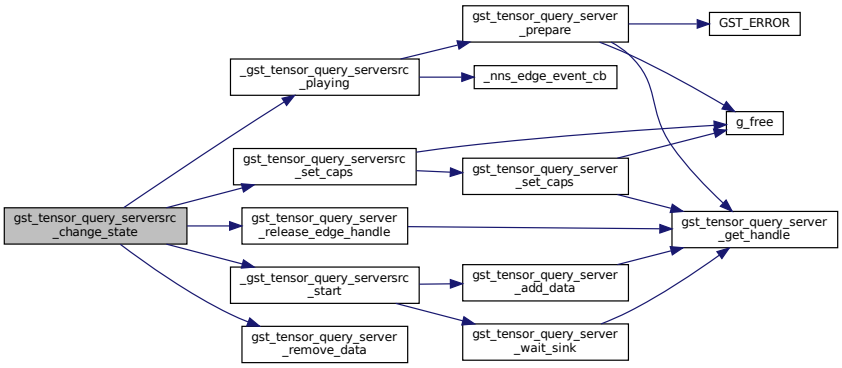
9.123.4.7 gst_tensor_query_serversrc_change_state()

```
static GstStateChangeReturn gst_tensor_query_serversrc_change_state (
            GstElement * element,
            GstStateChange transition )  [static]
```

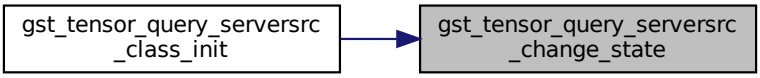
Change state of query server src.

Definition at line 275 of file tensor_query_serversrc.c.

Here is the call graph for this function:



Here is the caller graph for this function:



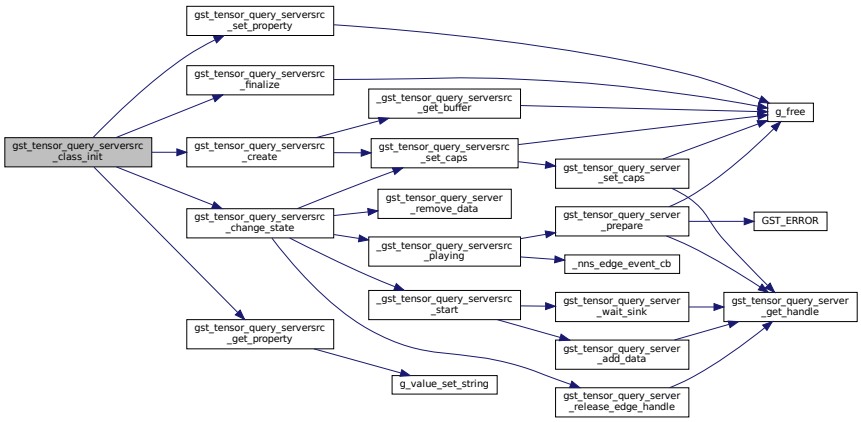
9.123.4.8 gst_tensor_query_serversrc_class_init()

```
static void gst_tensor_query_serversrc_class_init (
            GstTensorQueryServerSrcClass * klass )  [static]
```

initialize the query_serversrc class

Definition at line 77 of file tensor_query_serversrc.c.

Here is the call graph for this function:



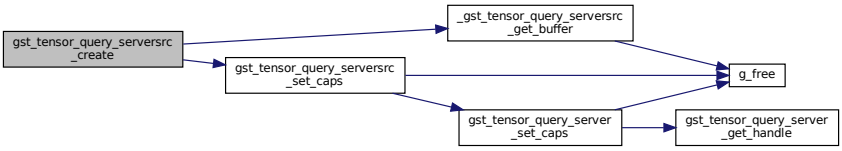
9.123.4.9 gst_tensor_query_serversrc_create()

```
static GstFlowReturn gst_tensor_query_serversrc_create (
            GstPushSrc * psrc,
            GstBuffer ** buf )  [static]
```

create query_serversrc, wait on socket and receive data

Definition at line 495 of file tensor_query_serversrc.c.

Here is the call graph for this function:



Here is the caller graph for this function:

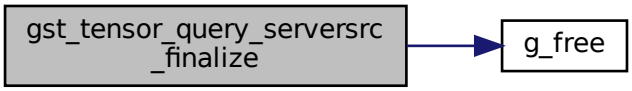
9.123.4.10 gst_tensor_query_serversrc_finalize()

```
static void gst_tensor_query_serversrc_finalize (
            GObject * object )  [static]
```

finalize the query_serversrc object

Definition at line 175 of file tensor_query_serversrc.c.

Here is the call graph for this function:



Here is the caller graph for this function:

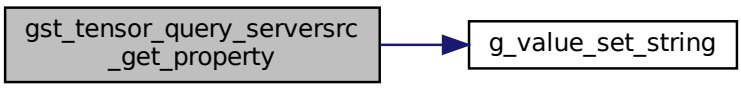
9.123.4.11 gst_tensor_query_serversrc_get_property()

```
static void gst_tensor_query_serversrc_get_property (
            GObject * object,
            guint prop_id,
            GValue * value,
            GParamSpec * pspec )  [static]
```

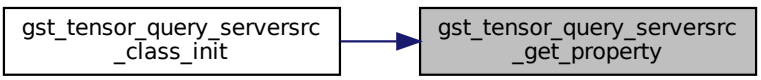
get property of query_serversrc

Definition at line 390 of file tensor_query_serversrc.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.123.4.12 gst_tensor_query_serversrc_init()

```
static void gst_tensor_query_serversrc_init (
            GstTensorQueryServerSrc * src )  [static]
```

initialize the new query_serversrc element

set the timestamps on each buffer

set the source to be live

Definition at line 150 of file tensor_query_serversrc.c.

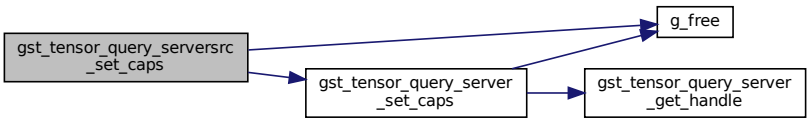
9.123.4.13 gst_tensor_query_serversrc_set_caps()

```
static gboolean gst_tensor_query_serversrc_set_caps (
            GstBaseSrc * bsrc,
            GstCaps * caps )  [static]
```

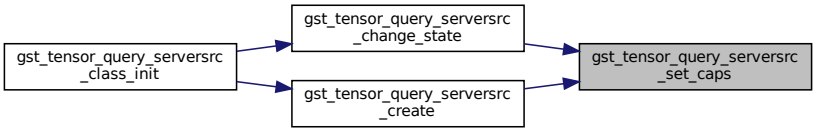
An implementation of the set_caps vmethod in GstBaseSrcClass.

Definition at line 533 of file tensor_query_serversrc.c.

Here is the call graph for this function:



Here is the caller graph for this function:



9.123.4.14 gst_tensor_query_serversrc_set_property()

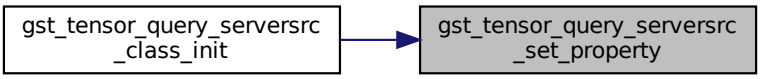
```
static void gst_tensor_query_serversrc_set_property (
            GObject * object,
            guint prop_id,
            const GValue * value,
            GParamSpec * pspec )  [static]
```

set property of query_serversrc

Definition at line 331 of file tensor_query_serversrc.c.

Here is the call graph for this function:

Here is the caller graph for this function:



9.123.5 Variable Documentation

9.123.5.1 srctemplate

```
GstStaticPadTemplate srctemplate  [static]
```

Initial value:

```
= GST_STATIC_PAD_TEMPLATE ("src",
    GST_PAD_SRC,
    GST_PAD_ALWAYS,
    GST_STATIC_CAPS_ANY)
```

the capabilities of the outputs

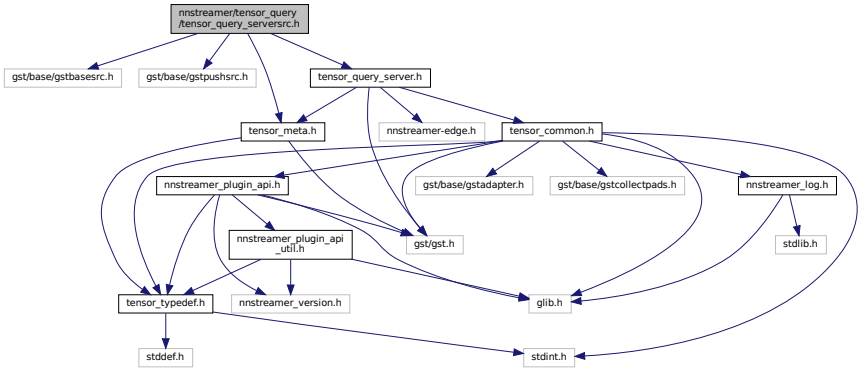
Definition at line 35 of file tensor_query_serversrc.c.

9.124 nnstreamer/tensor_query/tensor_query_serversrc.h File Reference

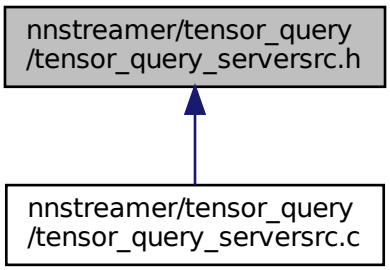
GStreamer plugin to handle tensor query_server src.

```
#include <gst/base/gstbasesrc.h>
#include <gst/base/gstpushsrc.h>
#include <tensor_meta.h>
#include "tensor_query_server.h"
```

Include dependency graph for tensor_query_serversrc.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct _GstTensorQueryServerSrc

  GstTensorQueryServerSrc data structure.
- struct _GstTensorQueryServerSrcClass

  GstTensorQueryServerSrcClass data structure.

Macros

- #define GST_TYPE_TENSOR_QUERY_SERVERSRC (gst_tensor_query_serversrc_get_type())
- #define GST_TENSOR_QUERY_SERVERSRC(obj) (G_TYPE_CHECK_INSTANCE_CAST((obj),GST_TYPE_TENSOR_QUE
- #define GST_TENSOR_QUERY_SERVERSRC_CLASS(klass) (G_TYPE_CHECK_CLASS_CAST((klass),GST_TYPE_TENS
- #define GST_IS_TENSOR_QUERY_SERVERSRC(obj) (G_TYPE_CHECK_INSTANCE_TYPE((obj),GST_TYPE_TENSOR_Q
- #define GST_IS_TENSOR_QUERY_SERVERSRC_CLASS(klass) (G_TYPE_CHECK_CLASS_TY↩ PE((klass),GST_TYPE_TENSOR_QUERY_SERVERSRC))
- #define GST_TENSOR_QUERY_SERVERSRC_CAST(obj) ((GstTensorQueryServerSrc *)(obj))

Typedefs

- typedef struct _GstTensorQueryServerSrc GstTensorQueryServerSrc
- typedef struct _GstTensorQueryServerSrcClass GstTensorQueryServerSrcClass

Functions

- GType gst_tensor_query_serversrc_get_type (void)

9.124.1 Detailed Description

GStreamer plugin to handle tensor query_server src.

Date

09 Jul 2021

Author

Junhwan Kim jejudo.kim@samsung.com

See also

http://github.com/nnstreamer/nnstreamer

Bug No known bugs

9.124.2 Macro Definition Documentation

9.124.2.1 GST_IS_TENSOR_QUERY_SERVERSRC

```
#define GST_IS_TENSOR_QUERY_SERVERSRC(
              obj ) (G_TYPE_CHECK_INSTANCE_TYPE((obj),GST_TYPE_TENSOR_QUERY_SERVERSRC))
```

Definition at line 28 of file tensor_query_serversrc.h.

9.124.2.2 GST_IS_TENSOR_QUERY_SERVERSRC_CLASS

```
#define GST_IS_TENSOR_QUERY_SERVERSRC_CLASS(
              klass ) (G_TYPE_CHECK_CLASS_TYPE((klass),GST_TYPE_TENSOR_QUERY_SERVERSRC))
```

Definition at line 30 of file tensor_query_serversrc.h.

9.124.2.3 GST_TENSOR_QUERY_SERVERSRC

```
#define GST_TENSOR_QUERY_SERVERSRC(
              obj ) (G_TYPE_CHECK_INSTANCE_CAST((obj),GST_TYPE_TENSOR_QUERY_SERVERSRC,GstTensorQueryServerSrc)
```

Definition at line 24 of file tensor_query_serversrc.h.

9.124.2.4 GST_TENSOR_QUERY_SERVERSRC_CAST

```
#define GST_TENSOR_QUERY_SERVERSRC_CAST(
              obj ) ((GstTensorQueryServerSrc *)(obj))
```

Definition at line 32 of file tensor_query_serversrc.h.

9.124.2.5 GST_TENSOR_QUERY_SERVERSRC_CLASS

```
#define GST_TENSOR_QUERY_SERVERSRC_CLASS(
              klass ) (G_TYPE_CHECK_CLASS_CAST((klass),GST_TYPE_TENSOR_QUERY_SERVERSRC,GstTensorQueryServerSrc
```

Definition at line 26 of file tensor_query_serversrc.h.

9.124.2.6 GST_TYPE_TENSOR_QUERY_SERVERSRC

```
#define GST_TYPE_TENSOR_QUERY_SERVERSRC (gst_tensor_query_serversrc_get_type())
```

Definition at line 22 of file tensor_query_serversrc.h.

9.124.3 Typedef Documentation

9.124.3.1 GstTensorQueryServerSrc

```
typedef struct _GstTensorQueryServerSrc GstTensorQueryServerSrc
```

Definition at line 33 of file tensor_query_serversrc.h.

9.124.3.2 GstTensorQueryServerSrcClass

```
typedef struct _GstTensorQueryServerSrcClass GstTensorQueryServerSrcClass
```

Definition at line 34 of file tensor_query_serversrc.h.

9.124.4 Function Documentation

9.124.4.1 gst_tensor_query_serversrc_get_type()

```
GType gst_tensor_query_serversrc_get_type (
            void  )
```

Index