4.5.2.7 ope_encoder_get_page() .................................................. 17
4.5.2.8 ope_encoder_drain() ..................................................... 18
4.5.2.9 ope_encoder_destroy() .................................................... 18
4.5.2.10 ope_encoder_chain_current() .......................................... 18
4.5.2.11 ope_encoder_continue_new_file() .................................... 19
4.5.2.12 ope_encoder_continue_new_callbacks() ............................. 19
4.5.2.13 ope_encoder_flush_header() ........................................... 19
4.5.2.14 ope_encoder_ctl() ..................................................... 20
4.5.2.15 ope_strerror() .......................................................... 20
4.5.2.16 ope_get_version_string() .............................................. 21
4.5.2.17 ope_get_abi_version() ................................................. 21

5 Data Structure Documentation .................................................. 23
5.1 OpusEncCallbacks Struct Reference ........................................ 23
  5.1.1 Detailed Description ..................................................... 23

Index ................................................................................. 25
Chapter 1

Main Page

1.1 Introduction

This is the documentation for the libopusenc C API.

The libopusenc package provides a convenient high-level API for encoding Ogg Opus files.

1.2 Organization

The main API is divided into several sections:

- Encoding
- Comments Handling
- Encoding Options
- Callback Functions
- Error Codes

1.3 Overview

The libopusfile API provides an easy way to encode Ogg Opus files using libopus.
# Module Index

## 2.1 Modules

Here is a list of all modules:

<table>
<thead>
<tr>
<th>Module</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error Codes</td>
<td>7</td>
</tr>
<tr>
<td>Encoding Options</td>
<td>8</td>
</tr>
<tr>
<td>Callback Functions</td>
<td>8</td>
</tr>
<tr>
<td>Comments Handling</td>
<td>10</td>
</tr>
<tr>
<td>Encoding</td>
<td>13</td>
</tr>
</tbody>
</table>
Chapter 3

Data Structure Index

3.1 Data Structures

Here are the data structures with brief descriptions:

OpusEncCallbacks
Callback functions for accessing the stream 23
Chapter 4

Module Documentation

4.1 Error Codes

List of possible error codes

Many of the functions in this library return a negative error code when a function fails. This list provides a brief explanation of the common errors. See each individual function for more details on what a specific error code means in that context.

- #define OPE_API_VERSION 0
  API version for this header.
- #define OPE_OK 0
- #define OPE_BAD_ARG -11
- #define OPE_INTERNAL_ERROR -13
- #define OPE_UNIMPLEMENTED -15
- #define OPE_ALLOC_FAIL -17
- #define OPE_CANNOT_OPEN -30
- #define OPE_TOO_LATE -31
- #define OPE_INVALID_PICTURE -32
- #define OPE_INVALID_ICON -33
- #define OPE_WRITE_FAIL -34
- #define OPE_CLOSE_FAIL -35

4.1.1 Detailed Description

4.1.2 Macro Definition Documentation

4.1.2.1 OPE_API_VERSION

#define OPE_API_VERSION 0

API version for this header.

Can be used to check for features at compile time.
4.2 Encoding Options

Control parameters

Macros for setting encoder options.

- \#define OPE_SET_DECISION_DELAY(x) OPE_SET_DECISION_DELAY_REQUEST, __opus_check_int(x)
- \#define OPE_GET_DECISION_DELAY(x) OPE_GET_DECISION_DELAY_REQUEST, __opus_check_int_ptr(x)
- \#define OPE_SET_MUXING_DELAY(x) OPE_SET_MUXING_DELAY_REQUEST, __opus_check_int_ptr(x)
- \#define OPE_GET_MUXING_DELAY(x) OPE_GET_MUXING_DELAY_REQUEST, __opus_check_int_ptr(x)
- \#define OPE_SET_COMMENT_PADDING(x) OPE_SET_COMMENT_PADDING_REQUEST, __opus_check_int_ptr(x)
- \#define OPE_GET_COMMENT_PADDING(x) OPE_GET_COMMENT_PADDING_REQUEST, __opus_check_int_ptr(x)
- \#define OPE_SET_SERIALNO(x) OPE_SET_SERIALNO_REQUEST, __opus_check_int(x)
- \#define OPE_GET_SERIALNO(x) OPE_GET_SERIALNO_REQUEST, __opus_check_int_ptr(x)
- \#define OPE_SET_PACKET_CALLBACK(x, u) OPE_SET_PACKET_CALLBACK_REQUEST, (x), (u)
- \#define OPE_SET_HEADER_GAIN(x) OPE_SET_HEADER_GAIN_REQUEST, __opus_check_int(x)
- \#define OPE_GET_HEADER_GAIN(x) OPE_GET_HEADER_GAIN_REQUEST, __opus_check_int_ptr(x)
- \#define OPE_GET_NB_STREAMS(x) OPE_GET_NB_STREAMS_REQUEST, __opus_check_int_ptr(x)
- \#define OPE_GET_NB_COUPLED_STREAMS(x) OPE_GET_NB_COUPLED_STREAMS_REQUEST, __opus_check_int_ptr(x)

4.2.1 Detailed Description

4.3 Callback Functions

Data Structures

- struct OpusEncCallbacks

  Callback functions for accessing the stream.

Callback functions

These are the callbacks that can be implemented for an encoder.

- typedef int(+ ope_write_func) (void +user_data, const unsigned char +ptr, opus_int32 len)
  Called for writing a page.
- typedef int(+ ope_close_func) (void +user_data)
  Called for closing a stream.
- typedef void(+ ope_packet_func) (void +user_data, const unsigned char +packet_ptr, opus_int32 packet_len, opus_uint32 flags)
  Called on every packet encoded (including header).
4.3 Callback Functions

4.3.1 Detailed Description

4.3.2 Typedef Documentation

4.3.2.1 ope_write_func

typedef int(* ope_write_func) (void *user_data, const unsigned char *ptr, opus_int32 len)

Called for writing a page.

Parameters

<table>
<thead>
<tr>
<th>parameter</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>user_data</td>
<td>user-defined data passed to the callback</td>
</tr>
<tr>
<td>ptr</td>
<td>buffer to be written</td>
</tr>
<tr>
<td>len</td>
<td>number of bytes to be written</td>
</tr>
</tbody>
</table>

Returns

error code

Return values

<table>
<thead>
<tr>
<th>value</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>success</td>
</tr>
<tr>
<td>1</td>
<td>failure</td>
</tr>
</tbody>
</table>

4.3.2.2 ope_close_func

typedef int(* ope_close_func) (void *user_data)

Called for closing a stream.

Parameters

<table>
<thead>
<tr>
<th>parameter</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>user_data</td>
<td>user-defined data passed to the callback</td>
</tr>
</tbody>
</table>

Returns

error code

Return values

<table>
<thead>
<tr>
<th>value</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>success</td>
</tr>
<tr>
<td>1</td>
<td>failure</td>
</tr>
</tbody>
</table>
4.3.2.3  ope_packet_func

typedef void(* ope_packet_func) (void *user_data, const unsigned char *packet_ptr, opus_int32 packet_len, opus_uint32 flags)

Called on every packet encoded (including header).

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>user_data</td>
<td>user-defined data passed to the callback</td>
</tr>
<tr>
<td>packet_ptr</td>
<td>packet data</td>
</tr>
<tr>
<td>packet_len</td>
<td>number of bytes in the packet</td>
</tr>
<tr>
<td>flags</td>
<td>optional flags (none defined for now so zero)</td>
</tr>
</tbody>
</table>

4.4  Comments Handling

Functions for handling comments

These functions make it possible to add comments and pictures to Ogg Opus files.

- OPE_EXPORT OggOpusComments * ope_comments_create (void)
  Create a new comments object.
- OPE_EXPORT OggOpusComments * ope_comments_copy (OggOpusComments *comments)
  Create a deep copy of a comments object.
- OPE_EXPORT void ope_comments_destroy (OggOpusComments *comments)
  Destroys a comments object.
- OPE_EXPORT int ope_comments_add (OggOpusComments *comments, const char *tag, const char *val)
  Add a comment.
- OPE_EXPORT int ope_comments_add_string (OggOpusComments *comments, const char *tag_and_val)
  Add a comment as a single tag=value string.
- OPE_EXPORT int ope_comments_add_picture (OggOpusComments *comments, const char *filename, int picture_type, const char *description)
  Add a picture from a file.
- OPE_EXPORT int ope_comments_add_picture_from_memory (OggOpusComments *comments, const char *ptr, size_t size, int picture_type, const char *description)
  Add a picture already in memory.

4.4.1  Detailed Description

4.4.2  Function Documentation
4.4 Comments Handling

4.4.2.1 ope_comments_create()

OPE_EXPORT OggOpusComments* ope_comments_create ( void )

Create a new comments object.

Returns

  Newly-created comments object.

4.4.2.2 ope_comments_copy()

OPE_EXPORT OggOpusComments* ope_comments_copy ( OggOpusComments * comments )

Create a deep copy of a comments object.

Parameters

  comments  Comments object to copy

Returns

  Deep copy of input.

4.4.2.3 ope_comments_destroy()

OPE_EXPORT void ope_comments_destroy ( OggOpusComments * comments )

Destroys a comments object.

Parameters

  comments  Comments object to destroy

4.4.2.4 ope_comments_add()

OPE_EXPORT int ope_comments_add ( OggOpusComments * comments,
### 4.4.2.5 ope_comments_add_string()

```c
OPE_EXPORT int ope_comments_add_string (  
    OggOpusComments * comments,  
    const char * tag_and_val  
)
```

Add a comment as a single tag=value string.

**Parameters**

<table>
<thead>
<tr>
<th>in, out</th>
<th>comments</th>
<th>Where to add the comments</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>tag</td>
<td>Tag for the comment (must not contain = char)</td>
</tr>
<tr>
<td></td>
<td>tag_and_val</td>
<td>string of the form tag=value (must contain = char)</td>
</tr>
</tbody>
</table>

**Returns**

Error code

### 4.4.2.6 ope_comments_add_picture()

```c
OPE_EXPORT int ope_comments_add_picture (  
    OggOpusComments * comments,  
    const char * filename,  
    int picture_type,  
    const char * description  
)
```

Add a picture from a file.

**Parameters**

<table>
<thead>
<tr>
<th>in, out</th>
<th>comments</th>
<th>Where to add the comments</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>filename</td>
<td>File name for the picture</td>
</tr>
<tr>
<td></td>
<td>picture_type</td>
<td>Type of picture (-1 for default)</td>
</tr>
<tr>
<td></td>
<td>description</td>
<td>Description (NULL means no comment)</td>
</tr>
</tbody>
</table>
4.5 Encoding

Returns

Error code

4.4.2.7  ope_comments_add_picture_from_memory()

OPE_EXPORT int ope_comments_add_picture_from_memory ( 
   OggOpusComments ∗ comments, 
   const char ∗ ptr, 
   size_t size, 
   int picture_type, 
   const char ∗ description )

Add a picture already in memory.

Parameters

<table>
<thead>
<tr>
<th>in,out</th>
<th>comments</th>
<th>Where to add the comments</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ptr</td>
<td>Pointer to picture in memory</td>
</tr>
<tr>
<td></td>
<td>size</td>
<td>Size of picture pointed to by ptr</td>
</tr>
<tr>
<td></td>
<td>picture_type</td>
<td>Type of picture (-1 for default)</td>
</tr>
<tr>
<td></td>
<td>description</td>
<td>Description (NULL means no comment)</td>
</tr>
</tbody>
</table>

Returns

Error code

4.5 Encoding

Functions for encoding Ogg Opus files

These functions make it possible to encode Ogg Opus files.

- OPE_EXPORT OggOpusEnc ∗ ope_encoder_create_file (const char ∗path, OggOpusComments ∗comments, opus_int32 rate, int channels, int family, int ∗error)
  
  Create a new OggOpus file.

- OPE_EXPORT OggOpusEnc ∗ ope_encoder_create_callbacks (const OpusEncCallbacks ∗callbacks, void ∗user_data, OggOpusComments ∗comments, opus_int32 rate, int channels, int family, int ∗error)
  
  Create a new OggOpus stream to be handled using callbacks.

- OPE_EXPORT OggOpusEnc ∗ ope_encoder_create_pull (OggOpusComments ∗comments, opus_int32 rate, int channels, int family, int ∗error)
  
  Create a new OggOpus stream to be used along with ope_encoder_get_page().

- OPE_EXPORT int ope_encoder_deferred_init_with_mapping (OggOpusEnc ∗enc, int family, int streams, int coupled_streams, const unsigned char ∗mapping)
  
  Deferred initialization of the encoder to force an explicit channel mapping.

- OPE_EXPORT int ope_encoder_write_float (OggOpusEnc ∗enc, const float ∗pcm, int samples_per_channel)
  
  Add/encode any number of float samples to the stream.
• OPE_EXPORT int ope_encoder_write (OggOpusEnc *enc, const opus_int16 *pcm, int samples_per_channel)

  Add/encode any number of 16-bit linear samples to the stream.

• OPE_EXPORT int ope_encoder_get_page (OggOpusEnc *enc, unsigned char **page, opus_int32 *len, int flush)

  Get the next page from the stream (only if using ope_encoder_create_pull()).

• OPE_EXPORT int ope_encoder_drain (OggOpusEnc *enc)

  Finalizes the stream, but does not deallocate the object.

• OPE_EXPORT void ope_encoder_destroy (OggOpusEnc *enc)

  Deallocates the object.

• OPE_EXPORT int ope_encoder_chain_current (OggOpusEnc *enc, OggOpusComments *comments)

  Ends the stream and create a new stream within the same file.

• OPE_EXPORT int ope_encoder_continue_new_file (OggOpusEnc *enc, const char *path, OggOpusComments *comments)

  Ends the stream and create a new file.

• OPE_EXPORT int ope_encoder_continue_new_callbacks (OggOpusEnc *enc, void *user_data, OggOpusComments *comments)

  Ends the stream and create a new file (callback-based).

• OPE_EXPORT int ope_encoder_flush_header (OggOpusEnc *enc)

  Write out the header now rather than wait for audio to begin.

• OPE_EXPORT int ope_encoder_ctl (OggOpusEnc *enc, int request,...)

  Sets encoder options.

• OPE_EXPORT const char * ope_strerror (int error)

  Converts a libopusenc error code into a human readable string.

• OPE_EXPORT const char * ope_get_version_string (void)

  Returns a string representing the version of libopusenc being used at run time.

• OPE_EXPORT int ope_get_abi_version (void)

  ABI version for this header.

4.5.1 Detailed Description

4.5.2 Function Documentation

4.5.2.1 ope_encoder_create_file()

OPE_EXPORT OggOpusEnc* ope_encoder_create_file {
  const char * path,
  OggOpusComments * comments,
  opus_int32 rate,
  int channels,
  int family,
  int * error
}

Create a new OggOpus file.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>path</td>
<td>Path where to create the file</td>
</tr>
<tr>
<td>comments</td>
<td>Comments associated with the stream</td>
</tr>
<tr>
<td>rate</td>
<td>Input sampling rate (48 kHz is faster)</td>
</tr>
<tr>
<td>channels</td>
<td>Number of channels</td>
</tr>
<tr>
<td>family</td>
<td>Mapping family (0 for mono/stereo, 1 for surround)</td>
</tr>
<tr>
<td>error</td>
<td>Error code (NULL if no error is to be returned)</td>
</tr>
</tbody>
</table>
4.5 Encoding

Returns
Newly-created encoder.

4.5.2.2 ope_encoder_create_callbacks()

OPE_EXPORT OggOpusEnc* ope_encoder_create_callbacks {
  const OpusEncCallbacks * callbacks,
  void * user_data,
  OggOpusComments * comments,
  opus_int32 rate,
  int channels,
  int family,
  int * error
}

Create a new OggOpus stream to be handled using callbacks.

Parameters

<table>
<thead>
<tr>
<th>callbacks</th>
<th>Callback functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>user_data</td>
<td>Pointer to be associated with the stream and passed to the callbacks</td>
</tr>
<tr>
<td>comments</td>
<td>Comments associated with the stream</td>
</tr>
<tr>
<td>rate</td>
<td>Input sampling rate (48 kHz is faster)</td>
</tr>
<tr>
<td>channels</td>
<td>Number of channels</td>
</tr>
<tr>
<td>family</td>
<td>Mapping family (0 for mono/stereo, 1 for surround)</td>
</tr>
<tr>
<td>out</td>
<td>error Error code (NULL if no error is to be returned)</td>
</tr>
</tbody>
</table>

Returns
Newly-created encoder.

4.5.2.3 ope_encoder_create_pull()

OPE_EXPORT OggOpusEnc* ope_encoder_create_pull {
  OggOpusComments * comments,
  opus_int32 rate,
  int channels,
  int family,
  int * error
}

Create a new OggOpus stream to be used along with ope_encoder_get_page().

This is mostly useful for muxing with other streams.

Parameters

<table>
<thead>
<tr>
<th>comments</th>
<th>Comments associated with the stream</th>
</tr>
</thead>
<tbody>
<tr>
<td>rate</td>
<td>Input sampling rate (48 kHz is faster)</td>
</tr>
<tr>
<td>channels</td>
<td>Number of channels</td>
</tr>
<tr>
<td>family</td>
<td>Mapping family (0 for mono/stereo, 1 for surround)</td>
</tr>
<tr>
<td>out</td>
<td>error Error code (NULL if no error is to be returned)</td>
</tr>
</tbody>
</table>
Returns

Newly-created encoder.

### 4.5.2.4 ope_encoder_deferred_init_with_mapping()

```c
OPE_EXPORT int ope_encoder_deferred_init_with_mapping (       
    OggOpusEnc * enc,                                          
    int family,                                                
    int streams,                                               
    int coupled_streams,                                       
    const unsigned char * mapping )                           
```

Deferred initialization of the encoder to force an explicit channel mapping.

This can be used to override the default channel coupling, but using it for regular surround will almost certainly lead to worse quality.

**Parameters**

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>in, out</strong></td>
<td><strong>enc</strong></td>
<td>Encoder</td>
</tr>
<tr>
<td></td>
<td><strong>family</strong></td>
<td>Mapping family (0 for mono/stereo, 1 for surround)</td>
</tr>
<tr>
<td></td>
<td><strong>streams</strong></td>
<td>Total number of streams</td>
</tr>
<tr>
<td></td>
<td><strong>coupled_streams</strong></td>
<td>Number of coupled streams</td>
</tr>
<tr>
<td></td>
<td><strong>mapping</strong></td>
<td>Channel mapping</td>
</tr>
</tbody>
</table>

**Returns**

Error code

### 4.5.2.5 ope_encoder_write_float()

```c
OPE_EXPORT int ope_encoder_write_float (       
    OggOpusEnc * enc,                                          
    const float * pcm,                                         
    int samples_per_channel )                                  
```

Add/encode any number of float samples to the stream.

**Parameters**

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>in, out</strong></td>
<td><strong>enc</strong></td>
<td>Encoder</td>
</tr>
<tr>
<td></td>
<td><strong>pcm</strong></td>
<td>Floating-point PCM values in the +/-1 range (interleaved if multiple channels)</td>
</tr>
<tr>
<td></td>
<td><strong>samples_per_channel</strong></td>
<td>Number of samples for each channel</td>
</tr>
</tbody>
</table>
4.5 Encoding

Returns

Error code

4.5.2.6 ope_encoder_write()

OPE_EXPORT int ope_encoder_write (  
    OggOpusEnc * enc,  
    const opus_int16 * pcm,  
    int samples_per_channel )

Add/encode any number of 16-bit linear samples to the stream.

Parameters

<table>
<thead>
<tr>
<th>in, out</th>
<th>enc</th>
<th>Encoder</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>pcm</td>
<td>Linear 16-bit PCM values in the [-32768,32767] range (interleaved if multiple channels)</td>
</tr>
<tr>
<td></td>
<td>samples_per_channel</td>
<td>Number of samples for each channel</td>
</tr>
</tbody>
</table>

Returns

Error code

4.5.2.7 ope_encoder_get_page()

OPE_EXPORT int ope_encoder_get_page (  
    OggOpusEnc * enc,  
    unsigned char ** page,  
    opus_int32 * len,  
    int flush )

Get the next page from the stream (only if using ope_encoder_create_pull()).

Parameters

<table>
<thead>
<tr>
<th>in, out</th>
<th>enc</th>
<th>Encoder</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>page</td>
<td>Next available encoded page</td>
</tr>
<tr>
<td></td>
<td>len</td>
<td>Size (in bytes) of the page returned</td>
</tr>
<tr>
<td></td>
<td>flush</td>
<td>If non-zero, forces a flush of the page (if any data available)</td>
</tr>
</tbody>
</table>

Returns

1 if there is a page available, 0 if not.
4.5.2.8  ope_encoder_drain()

OPE_EXPORT int ope_encoder_drain (  
    OggOpusEnc * enc  )

Finalizes the stream, but does not deallocate the object.

Parameters

    in, out  enc  Encoder

Returns

    Error code

4.5.2.9  ope_encoder_destroy()

OPE_EXPORT void ope_encoder_destroy (  
    OggOpusEnc * enc  )

Deallocates the object.

Make sure to ope_drain() first.

Parameters

    in, out  enc  Encoder

4.5.2.10  ope_encoder_chain_current()

OPE_EXPORT int ope_encoder_chain_current (  
    OggOpusEnc * enc,  
    OggOpusComments * comments  )

Ends the stream and create a new stream within the same file.

Parameters

    in, out  enc  Encoder

    comments  Comments associated with the stream

Returns

    Error code

Generated by Doxygen
4.5 Encoding

4.5.2.11 ope_encoder_continue_new_file()

OPE_EXPORT int ope_encoder_continue_new_file ( 
   OggOpusEnc * enc, 
   const char * path, 
   OggOpusComments * comments )

Ends the stream and create a new file.

Parameters

<table>
<thead>
<tr>
<th>in, out</th>
<th>enc</th>
<th>Encoder</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>path</td>
<td>Path where to write the new file</td>
</tr>
<tr>
<td></td>
<td>comments</td>
<td>Comments associated with the stream</td>
</tr>
</tbody>
</table>

Returns

Error code

4.5.2.12 ope_encoder_continue_new_callbacks()

OPE_EXPORT int ope_encoder_continue_new_callbacks ( 
   OggOpusEnc * enc, 
   void * user_data, 
   OggOpusComments * comments )

Ends the stream and create a new file (callback-based).

Parameters

<table>
<thead>
<tr>
<th>in, out</th>
<th>enc</th>
<th>Encoder</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>user_data</td>
<td>Pointer to be associated with the new stream and passed to the callbacks</td>
</tr>
<tr>
<td></td>
<td>comments</td>
<td>Comments associated with the stream</td>
</tr>
</tbody>
</table>

Returns

Error code

4.5.2.13 ope_encoder_flush_header()

OPE_EXPORT int ope_encoder_flush_header ( 
   OggOpusEnc * enc )

Write out the header now rather than wait for audio to begin.
Parameters

<table>
<thead>
<tr>
<th>in, out</th>
<th>enc</th>
<th>Encoder</th>
</tr>
</thead>
</table>

Returns

Error code

4.5.2.14 ope_encoder_ctl()

OPE_EXPORT int ope_encoder_ctl ( 
    OggOpusEnc * enc, 
    int request, 
    ... ) 

Sets encoder options.

Parameters

<table>
<thead>
<tr>
<th>in, out</th>
<th>enc</th>
<th>Encoder</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

Error code

4.5.2.15 ope_strerror()

OPE_EXPORT const char* ope_strerror ( 
    int error ) 

Converts a libopusenc error code into a human readable string.

Parameters

<table>
<thead>
<tr>
<th>error</th>
<th>Error number</th>
</tr>
</thead>
</table>

Returns

Error string
4.5.2.16 ope_get_version_string()

OPE_EXPORT const char* ope_get_version_string ( void )

Returns a string representing the version of libopusenc being used at run time.

Returns

A string describing the version of this library

4.5.2.17 ope_get_abi_version()

OPE_EXPORT int ope_get_abi_version ( void )

ABI version for this header.

Can be used to check for features at run time.

Returns

An integer representing the ABI version
Chapter 5

Data Structure Documentation

5.1 OpusEncCallbacks Struct Reference

Callback functions for accessing the stream.

#include <opusenc.h>

Data Fields

- ope_write_func write  
  Callback for writing to the stream.
- ope_close_func close  
  Callback for closing the stream.

5.1.1 Detailed Description

Callback functions for accessing the stream.

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

- opusenc.h
Index

Callback Functions, 8
  ope_close_func, 9
  ope_packet_func, 10
  ope_write_func, 9
Comments Handling, 10
  ope_comments_add, 11
  ope_comments_add_picture, 12
  ope_comments_add_picture_from_memory, 13
  ope_comments_add_string, 12
  ope_comments_copy, 11
  ope_comments_create, 10
  ope_comments_destroy, 11
Encoding, 13
  ope_encoder_chain_current, 18
  ope_encoder_continue_new_callbacks, 19
  ope_encoder_continue_new_file, 18
  ope_encoder_create_callbacks, 15
  ope_encoder_create_file, 14
  ope_encoder_create_pull, 15
  ope_encoder_ctl, 20
  ope_encoder_deferred_init_with_mapping, 16
  ope_encoder_destroy, 18
  ope_encoder_drain, 17
  ope_encoder_flush_header, 19
  ope_encoder_get_page, 17
  ope_encoder_write, 17
  ope_encoder_write_float, 16
  ope_get_abi_version, 21
  ope_get_version_string, 20
  ope_strerror, 20
Encoding Options, 8
Error Codes, 7
  OPE_API_VERSION, 7
OPE_API_VERSION
  Error Codes, 7
  ope_close_func
    Callback Functions, 9
  ope_comments_add
    Comments Handling, 11
  ope_comments_add_picture
    Comments Handling, 12
  ope_comments_add_picture_from_memory
    Comments Handling, 13
  ope_comments_add_string
    Comments Handling, 12
  ope_comments_copy
    Comments Handling, 11
  ope_comments_create
  Comments Handling, 10
  ope_comments_destroy
    Comments Handling, 11
  ope_encoder_chain_current
    Encoding, 18
  ope_encoder_continue_new_callbacks
    Encoding, 19
  ope_encoder_continue_new_file
    Encoding, 18
  ope_encoder_create_callbacks
    Encoding, 15
  ope_encoder_create_file
    Encoding, 14
  ope_encoder_create_pull
    Encoding, 15
  ope_encoder_ctl
    Encoding, 20
  ope_encoder_deferred_init_with_mapping
    Encoding, 16
  ope_encoder_destroy
    Encoding, 18
  ope_encoder_drain
    Encoding, 17
  ope_encoder_flush_header
    Encoding, 19
  ope_encoder_get_page
    Encoding, 17
  ope_encoder_write
    Encoding, 17
  ope_encoder_write_float
    Encoding, 16
  ope_get_abi_version
    Encoding, 21
  ope_strerror
    Encoding, 20
  ope_packet_func
    Callback Functions, 10
  ope_strerror
    Encoding, 20
  ope_write_func
    Callback Functions, 9
OpusEncCallbacks, 23