libopusenc

0.2.1-13-gdc6ab59

Generated by Doxygen 1.9.1

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`.

Chapter 2

Module Index

2.1 Modules

Here is a list of all modules:

Chapter 3

Data Structure Index

3.1 Data Structures

Here are the data structures with brief descriptions:

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(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(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).

4.3.1 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

user_data	user-defined data passed to the callback
ptr	buffer to be written
len	number of bytes to be written

Returns

error code

Return values

0	success
1	failure

4.3.2.2 ope_close_func

```
typedef int(* ope_close_func) (void *user_data)
```

Called for closing a stream.

Parameters

user_data	user-defined data passed to the callback

Returns

error code

Return values

0	success
1	failure

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

user_data	user-defined data passed to the callback
packet_ptr	packet data
packet_len	number of bytes in the packet
flags	optional flags (none defined for now so zero)

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

4.4.2.1 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,
```

```
            const char * tag,
            const char * val )
```

Add a comment.

Parameters

in,out		
	comments	Where to add the comments
	tag	Tag for the comment (must not contain = char)
	val	Value for the tag

Returns

Error code

4.4.2.5 ope_comments_add_string()

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

Add a comment as a single tag=value string.

Parameters

in,out		
	comments	Where to add the comments
	tag_and_val	string of the form tag=value (must contain = char)

Returns

Error code

4.4.2.6 ope_comments_add_picture()

```
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

in,out		
	comments	Where to add the comments
	filename	File name for the picture
	picture_type	Type of picture (-1 for default)
	description	Description (NULL means no comment)

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

in,out	*comments*	Where to add the comments
	ptr	Pointer to picture in memory
	size	Size of picture pointed to by ptr
	picture_type	Type of picture (-1 for default)
	description	Description (NULL means no comment)

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 obect.
- 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, OggOpus↩ Comments ∗comments)

  Ends the stream and create a new file.
- OPE_EXPORT int ope_encoder_continue_new_callbacks (OggOpusEnc ∗enc, void ∗user_data, OggOpus↩ Comments ∗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

	path	Path where to create the file
	comments	Comments associated with the stream
	rate	Input sampling rate (48 kHz is faster)
	channels	Number of channels
	family	Mapping family (0 for mono/stereo, 1 for surround)
out	*error*	Error code (NULL if no error is to be returned)

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

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

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

	comments	Comments associated with the stream
	rate	Input sampling rate (48 kHz is faster)
	channels	Number of channels
	family	Mapping family (0 for mono/stereo, 1 for surround)
out	*error*	Error code (NULL if no error is to be returned)

Returns

Newly-created encoder.

4.5.2.4 ope_encoder_deferred_init_with_mapping()

```
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

in,out		
	enc	Encoder
	family	Mapping family (0 for mono/stereo, 1 for surround)
	streams	Total number of streams
	coupled_streams	Number of coupled streams
	mapping	Channel mapping

Returns

Error code

4.5.2.5 ope_encoder_write_float()

```
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

in,out		
	enc	Encoder
	pcm	Floating-point PCM values in the +/-1 range (interleaved if multiple channels)
	samples_per_channel	Number of samples for each channel

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

in,out	*enc*	Encoder
	pcm	Linear 16-bit PCM values in the [-32768,32767] range (interleaved if multiple channels)
	samples_per_channel	Number of samples for each channel

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

in,out	*enc*	Encoder
out	*page*	Next available encoded page
out	*len*	Size (in bytes) of the page returned
	flush	If non-zero, forces a flush of the page (if any data avaiable)

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 obect.

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

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

in,out		
	enc	Encoder
	path	Path where to write the new file
	comments	Comments associated with the stream

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

in,out		
	enc	Encoder
	user_data	Pointer to be associated with the new stream and passed to the callbacks
	comments	Comments associated with the stream

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

in,out	*enc*	Encoder

Returns

Error code

4.5.2.14 ope_encoder_ctl()

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

Sets encoder options.

Parameters

in,out	*enc*	Encoder
	request	Use a request macro

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

error	Error number

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