

Opus

0.9.6-149-g48069bf

Generated by Doxygen 1.7.4

Fri Sep 16 2011 00:51:57



# Contents

<b>1</b>	<b>Module Index</b>	<b>1</b>
1.1	Modules . . . . .	1
<b>2</b>	<b>File Index</b>	<b>3</b>
2.1	File List . . . . .	3
<b>3</b>	<b>Module Documentation</b>	<b>5</b>
3.1	Opus Encoder . . . . .	5
3.1.1	Typedef Documentation . . . . .	6
3.1.1.1	OpusEncoder . . . . .	6
3.1.2	Function Documentation . . . . .	6
3.1.2.1	opus_encode . . . . .	6
3.1.2.2	opus_encode_float . . . . .	6
3.1.2.3	opus_encoder_create . . . . .	7
3.1.2.4	opus_encoder_ctl . . . . .	7
3.1.2.5	opus_encoder_destroy . . . . .	8
3.1.2.6	opus_encoder_get_size . . . . .	8
3.1.2.7	opus_encoder_init . . . . .	8
3.2	Opus Decoder . . . . .	8
3.2.1	Typedef Documentation . . . . .	9
3.2.1.1	OpusDecoder . . . . .	9
3.2.2	Function Documentation . . . . .	10
3.2.2.1	opus_decode . . . . .	10
3.2.2.2	opus_decode_float . . . . .	10
3.2.2.3	opus_decoder_create . . . . .	11
3.2.2.4	opus_decoder_ctl . . . . .	11

3.2.2.5	<a href="#">opus_decoder_destroy</a>	11
3.2.2.6	<a href="#">opus_decoder_get_nb_samples</a>	11
3.2.2.7	<a href="#">opus_decoder_get_size</a>	12
3.2.2.8	<a href="#">opus_decoder_init</a>	12
3.2.2.9	<a href="#">opus_packet_get_bandwidth</a>	12
3.2.2.10	<a href="#">opus_packet_get_nb_channels</a>	13
3.2.2.11	<a href="#">opus_packet_get_nb_frames</a>	13
3.2.2.12	<a href="#">opus_packet_get_samples_per_frame</a>	14
3.2.2.13	<a href="#">opus_packet_parse</a>	14
3.3	Repacketizer	14
3.3.1	Typedef Documentation	15
3.3.1.1	<a href="#">OpusRepacketizer</a>	15
3.3.2	Function Documentation	15
3.3.2.1	<a href="#">opus_repacketizer_cat</a>	15
3.3.2.2	<a href="#">opus_repacketizer_create</a>	15
3.3.2.3	<a href="#">opus_repacketizer_destroy</a>	15
3.3.2.4	<a href="#">opus_repacketizer_get_nb_frames</a>	15
3.3.2.5	<a href="#">opus_repacketizer_get_size</a>	15
3.3.2.6	<a href="#">opus_repacketizer_init</a>	15
3.3.2.7	<a href="#">opus_repacketizer_out</a>	15
3.3.2.8	<a href="#">opus_repacketizer_out_range</a>	15
3.4	Error codes	15
3.4.1	Define Documentation	16
3.4.1.1	<a href="#">OPUS_ALLOC_FAIL</a>	16
3.4.1.2	<a href="#">OPUS_BAD_ARG</a>	16
3.4.1.3	<a href="#">OPUS_BUFFER_TOO_SMALL</a>	16
3.4.1.4	<a href="#">OPUS_INTERNAL_ERROR</a>	16
3.4.1.5	<a href="#">OPUS_INVALID_PACKET</a>	16
3.4.1.6	<a href="#">OPUS_INVALID_STATE</a>	16
3.4.1.7	<a href="#">OPUS_OK</a>	16
3.4.1.8	<a href="#">OPUS_UNIMPLEMENTED</a>	17
3.5	Encoder related CTLs	17
3.5.1	Detailed Description	18
3.5.2	Define Documentation	18

3.5.2.1	OPUS_GET_APPLICATION . . . . .	18
3.5.2.2	OPUS_GET_BITRATE . . . . .	18
3.5.2.3	OPUS_GET_COMPLEXITY . . . . .	19
3.5.2.4	OPUS_GET_DTX . . . . .	19
3.5.2.5	OPUS_GET_FORCE_CHANNELS . . . . .	19
3.5.2.6	OPUS_GET_INBAND_FEC . . . . .	19
3.5.2.7	OPUS_GET_LOOKAHEAD . . . . .	20
3.5.2.8	OPUS_GET_PACKET_LOSS_PERC . . . . .	20
3.5.2.9	OPUS_GET_RESTRICTED_LOWDELAY . . . . .	20
3.5.2.10	OPUS_GET_SIGNAL . . . . .	21
3.5.2.11	OPUS_GET_VBR . . . . .	21
3.5.2.12	OPUS_GET_VBR_CONSTRAINT . . . . .	21
3.5.2.13	OPUS_GET_VOICE_RATIO . . . . .	21
3.5.2.14	OPUS_SET_APPLICATION . . . . .	22
3.5.2.15	OPUS_SET_BANDWIDTH . . . . .	22
3.5.2.16	OPUS_SET_BITRATE . . . . .	22
3.5.2.17	OPUS_SET_COMPLEXITY . . . . .	23
3.5.2.18	OPUS_SET_DTX . . . . .	23
3.5.2.19	OPUS_SET_FORCE_CHANNELS . . . . .	23
3.5.2.20	OPUS_SET_INBAND_FEC . . . . .	23
3.5.2.21	OPUS_SET_PACKET_LOSS_PERC . . . . .	24
3.5.2.22	OPUS_SET_RESTRICTED_LOWDELAY . . . . .	24
3.5.2.23	OPUS_SET_SIGNAL . . . . .	24
3.5.2.24	OPUS_SET_VBR . . . . .	24
3.5.2.25	OPUS_SET_VBR_CONSTRAINT . . . . .	25
3.5.2.26	OPUS_SET_VOICE_RATIO . . . . .	25
3.6	Generic CTLs . . . . .	25
3.6.1	Detailed Description . . . . .	26
3.6.2	Define Documentation . . . . .	26
3.6.2.1	OPUS_GET_BANDWIDTH . . . . .	26
3.6.2.2	OPUS_GET_FINAL_RANGE . . . . .	26
3.6.2.3	OPUS_GET_PITCH . . . . .	26
3.6.2.4	OPUS_RESET_STATE . . . . .	27
3.7	Opus library information functions . . . . .	27

3.7.1	Function Documentation	27
3.7.1.1	opus_get_version_string	27
3.7.1.2	opus_strerror	27
<b>4</b>	<b>File Documentation</b>	<b>29</b>
4.1	opus.h File Reference	29
4.1.1	Detailed Description	31
4.2	opus_defines.h File Reference	31
4.2.1	Detailed Description	33
4.3	opus_multistream.h File Reference	33
4.3.1	Detailed Description	34
4.3.2	Define Documentation	35
4.3.2.1	__opus_check_decstate_ptr	35
4.3.2.2	__opus_check_encstate_ptr	35
4.3.2.3	OPUS_MULTISTREAM_GET_DECODER_STATE	35
4.3.2.4	OPUS_MULTISTREAM_GET_DECODER_STATE_REQUEST	35
4.3.2.5	OPUS_MULTISTREAM_GET_ENCODER_STATE	35
4.3.2.6	OPUS_MULTISTREAM_GET_ENCODER_STATE_REQUEST	35
4.3.3	Typedef Documentation	35
4.3.3.1	OpusMSDecoder	35
4.3.3.2	OpusMSEncoder	35
4.3.4	Function Documentation	35
4.3.4.1	opus_multistream_decode	35
4.3.4.2	opus_multistream_decode_float	35
4.3.4.3	opus_multistream_decoder_create	36
4.3.4.4	opus_multistream_decoder_ctl	36
4.3.4.5	opus_multistream_decoder_destroy	36
4.3.4.6	opus_multistream_decoder_init	36
4.3.4.7	opus_multistream_encode	37
4.3.4.8	opus_multistream_encode_float	37
4.3.4.9	opus_multistream_encoder_create	37
4.3.4.10	opus_multistream_encoder_ctl	38
4.3.4.11	opus_multistream_encoder_destroy	38
4.3.4.12	opus_multistream_encoder_init	38

---

4.4	opus_types.h File Reference . . . . .	38
4.4.1	Detailed Description . . . . .	39
4.4.2	Define Documentation . . . . .	39
4.4.2.1	opus_int . . . . .	39
4.4.2.2	opus_int64 . . . . .	39
4.4.2.3	opus_int8 . . . . .	39
4.4.2.4	opus_uint . . . . .	39
4.4.2.5	opus_uint64 . . . . .	39
4.4.2.6	opus_uint8 . . . . .	39
4.4.3	Typedef Documentation . . . . .	39
4.4.3.1	opus_int16 . . . . .	39
4.4.3.2	opus_int32 . . . . .	39
4.4.3.3	opus_uint16 . . . . .	39
4.4.3.4	opus_uint32 . . . . .	39





# Chapter 1

## Module Index

### 1.1 Modules

Here is a list of all modules:

Opus Encoder . . . . .	<a href="#">5</a>
Opus Decoder . . . . .	<a href="#">8</a>
Repacketizer . . . . .	<a href="#">14</a>
Error codes . . . . .	<a href="#">15</a>
Encoder related CTLs . . . . .	<a href="#">17</a>
Generic CTLs . . . . .	<a href="#">25</a>
Opus library information functions . . . . .	<a href="#">27</a>



## Chapter 2

# File Index

### 2.1 File List

Here is a list of all files with brief descriptions:

<a href="#">opus.h</a> (Opus reference implementation API ) . . . . .	29
<a href="#">opus_defines.h</a> (Opus reference implementation constants ) . . . . .	31
<a href="#">opus_multistream.h</a> (Opus reference implementation multistream API ) . . . .	33
<a href="#">opus_types.h</a> (Opus reference implementation types ) . . . . .	38



## Chapter 3

# Module Documentation

### 3.1 Opus Encoder

#### Typedefs

- typedef struct [OpusEncoder](#) [OpusEncoder](#)  
*Opus encoder state.*

#### Functions

- int [opus\\_encoder\\_get\\_size](#) (int channels)
- [OpusEncoder](#) \* [opus\\_encoder\\_create](#) ([opus\\_int32](#) Fs, int channels, int application, int \*error)  
*Allocates and initializes an encoder state.*
- int [opus\\_encoder\\_init](#) ([OpusEncoder](#) \*st, [opus\\_int32](#) Fs, int channels, int application)  
*Initializes a previously allocated encoder state. The memory pointed to by st must be the size returned by opus\_encoder\_get\_size.*
- int [opus\\_encode](#) ([OpusEncoder](#) \*st, const [opus\\_int16](#) \*pcm, int frame\_size, unsigned char \*data, int max\_data\_bytes)  
*Encodes an Opus frame.*
- int [opus\\_encode\\_float](#) ([OpusEncoder](#) \*st, const float \*pcm, int frame\_size, unsigned char \*data, int max\_data\_bytes)  
*Encodes an Opus frame from floating point input.*
- void [opus\\_encoder\\_destroy](#) ([OpusEncoder](#) \*st)  
*Frees an OpusEncoder allocated by opus\_encoder\_create.*
- int [opus\\_encoder\\_ctl](#) ([OpusEncoder](#) \*st, int request,...)  
*Perform a CTL function on an Opus encoder.*

### 3.1.1 Typedef Documentation

#### 3.1.1.1 typedef struct OpusEncoder OpusEncoder

Opus encoder state.

This contains the complete state of an Opus encoder. It is position independent and can be freely copied.

#### See also

[opus\\_encoder\\_create](#), [opus\\_encoder\\_init](#)

### 3.1.2 Function Documentation

#### 3.1.2.1 int opus\_encode ( OpusEncoder \* *st*, const opus\_int16 \* *pcm*, int *frame\_size*, unsigned char \* *data*, int *max\_data\_bytes* )

Encodes an Opus frame.

The passed *frame\_size* must be an opus frame size for the encoder's sampling rate. For example, at 48kHz the permitted values are 120, 240, 480, or 960. Passing in a duration of less than 10ms (480 samples at 48kHz) will prevent the encoder from using the LPC or hybrid modes.

#### Parameters

in	<i>st</i>	OpusEncoder*: Encoder state
in	<i>pcm</i>	opus_int16*: Input signal (interleaved if 2 channels). length is <i>frame_size</i> *channels*sizeof(opus_int16)
in	<i>frame_size</i>	int: Number of samples per frame of input signal
out	<i>data</i>	char*: Output payload (at least <i>max_data_bytes</i> long)
in	<i>max_data_bytes</i>	int: Allocated memory for payload; don't use for controlling bitrate

#### Returns

length of the data payload (in bytes)

#### 3.1.2.2 int opus\_encode\_float ( OpusEncoder \* *st*, const float \* *pcm*, int *frame\_size*, unsigned char \* *data*, int *max\_data\_bytes* )

Encodes an Opus frame from floating point input.

The passed *frame\_size* must be an opus frame size for the encoder's sampling rate. For example, at 48kHz the permitted values are 120, 240, 480, or 960. Passing in a duration of less than 10ms (480 samples at 48kHz) will prevent the encoder from using the LPC or hybrid modes.

#### Parameters

in	<i>st</i>	OpusEncoder*: Encoder state
in	<i>pcm</i>	float*: Input signal (interleaved if 2 channels). length is frame_size*channels*sizeof(float)
in	<i>frame_size</i>	int: Number of samples per frame of input signal
out	<i>data</i>	char*: Output payload (at least max_data_bytes long)
in	<i>max_data_bytes</i>	int: Allocated memory for payload; don't use for controlling bitrate

**Returns**

length of the data payload (in bytes)

### 3.1.2.3 OpusEncoder\* opus\_encoder\_create ( opus\_int32 *Fs*, int *channels*, int *application*, int \* *error* )

Allocates and initializes an encoder state.

There are three coding modes: OPUS\_APPLICATION\_VOIP gives best quality at a given bitrate for voice signals. It enhances the input signal by high-pass filtering and emphasizing formants and harmonics. Optionally it includes in-band forward error correction to protect against packet loss. Use this mode for typical VoIP applications. Because of the enhancement, even at high bitrates the output may sound different from the input. OPUS\_APPLICATION\_AUDIO gives best quality at a given bitrate for most non-voice signals like music. Use this mode for music and mixed (music/voice) content, broadcast, and applications requiring less than 15 ms of coding delay. OPUS\_APPLICATION\_RESTRICTED\_LOWDELAY configures low-delay mode that disables the speech-optimized mode in exchange for slightly reduced delay. This is useful when the caller knows that the speech-optimized modes will not be needed (use with caution).

**Parameters**

in	<i>Fs</i>	opus_int32: Sampling rate of input signal (Hz)
in	<i>channels</i>	int: Number of channels (1/2) in input signal
in	<i>application</i>	int: Coding mode (OPUS_APPLICATION_VOIP/OPUS_APPLICATION_AUDIO/OPUS_APPLICATION_RESTRICTED_LOWDELAY)
out	<i>error</i>	int*: Error code

### 3.1.2.4 int opus\_encoder\_ctl ( OpusEncoder \* *st*, int *request*, ... )

Perform a CTL function on an Opus encoder.

**See also**

[Encoder related CTLs](#)

### 3.1.2.5 void opus\_encoder\_destroy ( OpusEncoder \* st )

Frees an OpusEncoder allocated by opus\_encoder\_create.

#### Parameters

in	st	OpusEncoder*: State to be freed.
----	----	----------------------------------

### 3.1.2.6 int opus\_encoder\_get\_size ( int channels )

### 3.1.2.7 int opus\_encoder\_init ( OpusEncoder \* st, opus\_int32 Fs, int channels, int application )

Initializes a previously allocated encoder state. The memory pointed to by st must be the size returned by opus\_encoder\_get\_size.

This is intended for applications which use their own allocator instead of malloc.

#### See also

[opus\\_encoder\\_create](#), [opus\\_encoder\\_get\\_size](#) To reset a previously initialized state use the [OPUS\\_RESET\\_STATE](#) CTL.

#### Parameters

in	st	OpusEncoder*: Encoder state
in	Fs	opus_int32: Sampling rate of input signal (Hz)
in	channels	int: Number of channels (1/2) in input signal
in	application	int: Coding mode (OPUS_APPLICATION_VOIP/OPUS_APPLICATION_AUDIO/OPUS_APPLICATION_RESTRICTED_LOWDELAY)

#### Return values

OPUS_OK	Success.
---------	----------

## 3.2 Opus Decoder

#### Typedefs

- typedef struct [OpusDecoder](#) [OpusDecoder](#)  
*Opus decoder state.*

#### Functions

- int [opus\\_decoder\\_get\\_size](#) (int channels)  
*Gets the size of an OpusDecoder structure.*



- [OpusDecoder](#) \* [opus\\_decoder\\_create](#) ([opus\\_int32](#) Fs, int channels, int \*error)  
*Allocates and initializes a decoder state.*
- int [opus\\_decoder\\_init](#) ([OpusDecoder](#) \*st, [opus\\_int32](#) Fs, int channels)  
*Initializes a previously allocated decoder state.*
- int [opus\\_decode](#) ([OpusDecoder](#) \*st, const unsigned char \*data, int len, [opus\\_int16](#) \*pcm, int frame\_size, int decode\_fec)  
*Decode an Opus frame.*
- int [opus\\_decode\\_float](#) ([OpusDecoder](#) \*st, const unsigned char \*data, int len, float \*pcm, int frame\_size, int decode\_fec)  
*Decode an opus frame with floating point output.*
- int [opus\\_decoder\\_ctl](#) ([OpusDecoder](#) \*st, int request,...)  
*Perform a CTL function on an Opus decoder.*
- void [opus\\_decoder\\_destroy](#) ([OpusDecoder](#) \*st)  
*Frees an OpusDecoder allocated by [opus\\_decoder\\_create](#).*
- int [opus\\_packet\\_parse](#) (const unsigned char \*data, int len, unsigned char \*out\_toc, const unsigned char \*frames[48], short size[48], int \*payload\_offset)  
*Parse an opus packet into one or more frames.*
- int [opus\\_packet\\_get\\_bandwidth](#) (const unsigned char \*data)  
*Gets the bandwidth of an Opus packet.*
- int [opus\\_packet\\_get\\_samples\\_per\\_frame](#) (const unsigned char \*data, [opus\\_int32](#) Fs)  
*Gets the number of samples per frame from an Opus packet.*
- int [opus\\_packet\\_get\\_nb\\_channels](#) (const unsigned char \*data)  
*Gets the number of channels from an Opus packet.*
- int [opus\\_packet\\_get\\_nb\\_frames](#) (const unsigned char packet[], int len)  
*Gets the number of frame in an Opus packet.*
- int [opus\\_decoder\\_get\\_nb\\_samples](#) (const [OpusDecoder](#) \*dec, const unsigned char packet[], int len)  
*Gets the number of samples of an Opus packet.*

### 3.2.1 Typedef Documentation

#### 3.2.1.1 typedef struct OpusDecoder OpusDecoder

Opus decoder state.

This contains the complete state of an Opus decoder. It is position independent and can be freely copied.

#### See also

[opus\\_decoder\\_create](#), [opus\\_decoder\\_init](#)

### 3.2.2 Function Documentation

3.2.2.1 `int opus_decode ( OpusDecoder * st, const unsigned char * data, int len, opus_int16 * pcm, int frame_size, int decode_fec )`

Decode an Opus frame.

#### Parameters

in	<i>st</i>	OpusDecoder*: Decoder state
in	<i>data</i>	char*: Input payload. Use a NULL pointer to indicate packet loss
in	<i>len</i>	int: Number of bytes in payload*
out	<i>pcm</i>	opus_int16*: Output signal (interleaved if 2 channels). length is frame_size*channels*sizeof(opus_int16)
in	<i>frame_size</i>	Number of samples per channel of available space in *pcm, if less than the maximum frame size (120ms) some frames can not be decoded
in	<i>decode_fec</i>	int: Flag (0/1) to request that any in-band forward error correction data be decoded. If no such data is available the frame is decoded as if it were lost.

#### Returns

Number of decoded samples

3.2.2.2 `int opus_decode_float ( OpusDecoder * st, const unsigned char * data, int len, float * pcm, int frame_size, int decode_fec )`

Decode an opus frame with floating point output.

#### Parameters

in	<i>st</i>	OpusDecoder*: Decoder state
in	<i>data</i>	char*: Input payload. Use a NULL pointer to indicate packet loss
in	<i>len</i>	int: Number of bytes in payload
out	<i>pcm</i>	float*: Output signal (interleaved if 2 channels). length is frame_size*channels*sizeof(float)
in	<i>frame_size</i>	Number of samples per channel of available space in *pcm, if less than the maximum frame size (120ms) some frames can not be decoded
in	<i>decode_fec</i>	int: Flag (0/1) to request that any in-band forward error correction data be decoded. If no such data is available the frame is decoded as if it were lost.

#### Returns

Number of decoded samples

### 3.2.2.3 OpusDecoder\* opus\_decoder\_create ( opus\_int32 *Fs*, int *channels*, int \* *error* )

Allocates and initializes a decoder state.

#### Parameters

in	<i>Fs</i>	opus_int32: Sampling rate of input signal (Hz)
in	<i>channels</i>	int: Number of channels (1/2) in input signal
out	<i>error</i>	int*: Error code

### 3.2.2.4 int opus\_decoder\_ctl ( OpusDecoder \* *st*, int *request*, ... )

Perform a CTL function on an Opus decoder.

#### See also

decoderctls

### 3.2.2.5 void opus\_decoder\_destroy ( OpusDecoder \* *st* )

Frees an OpusDecoder allocated by opus\_decoder\_create.

#### Parameters

in	<i>st</i>	OpusDecoder*: State to be freed.
----	-----------	----------------------------------

### 3.2.2.6 int opus\_decoder\_get\_nb\_samples ( const OpusDecoder \* *dec*, const unsigned char *packet*[], int *len* )

Gets the number of samples of an Opus packet.

#### Parameters

in	<i>dec</i>	OpusDecoder*: Decoder state
in	<i>packet</i>	char*: Opus packet
in	<i>len</i>	int: Length of packet

#### Returns

Number of samples

#### Return values

<i>OPUS_INVALID_PACKET</i>	The compressed data passed is corrupted or of an unsupported type
----------------------------	---

### 3.2.2.7 `int opus_decoder_get_size ( int channels )`

Gets the size of an OpusDecoder structure.

#### Parameters

in	<i>channels</i>	int: Number of channels
----	-----------------	-------------------------

#### Returns

size

### 3.2.2.8 `int opus_decoder_init ( OpusDecoder * st, opus_int32 Fs, int channels )`

Initializes a previously allocated decoder state.

The state must be the size returned by `opus_decoder_get_size`. This is intended for applications which use their own allocator instead of `malloc`.

#### See also

[opus\\_decoder\\_create](#), [opus\\_decoder\\_get\\_size](#) To reset a previously initialized state use the `OPUS_RESET_STATE` CTL.

#### Parameters

in	<i>st</i>	OpusDecoder*: Decoder state.
in	<i>Fs</i>	opus_int32: Sampling rate of input signal (Hz)
in	<i>channels</i>	int: Number of channels (1/2) in input signal

#### Return values

<i>OPUS_OK</i>	Success.
----------------	----------

### 3.2.2.9 `int opus_packet_get_bandwidth ( const unsigned char * data )`

Gets the bandwidth of an Opus packet.

#### Parameters

in	<i>data</i>	char*: Opus packet
----	-------------	--------------------

#### Return values

<i>OPUS_BANDWIDTH_NARROWBAND</i>	Narrowband (4kHz bandpass)
<i>OPUS_BANDWIDTH_MEDIUMBAND</i>	Mediumband (6kHz bandpass)

<i>OPUS_ - BANDWIDTH_ - WIDEBAND</i>	Wideband (8kHz bandpass)
<i>OPUS_ - BANDWIDTH_ - SUPERWIDEBAND</i>	Superwideband (12kHz bandpass)
<i>OPUS_ - BANDWIDTH_ - FULLBAND</i>	Fullband (20kHz bandpass)
<i>OPUS_INVALID_ - PACKET</i>	The compressed data passed is corrupted or of an unsupported type

#### 3.2.2.10 int opus\_packet\_get\_nb\_channels ( const unsigned char \* *data* )

Gets the number of channels from an Opus packet.

##### Parameters

in	<i>data</i>	char*: Opus packet
----	-------------	--------------------

##### Returns

Number of channels

##### Return values

<i>OPUS_INVALID_ - PACKET</i>	The compressed data passed is corrupted or of an unsupported type
-----------------------------------	---

#### 3.2.2.11 int opus\_packet\_get\_nb\_frames ( const unsigned char *packet*[], int *len* )

Gets the number of frame in an Opus packet.

##### Parameters

in	<i>packet</i>	char*: Opus packet
in	<i>len</i>	int: Length of packet

##### Returns

Number of frames

##### Return values

<i>OPUS_INVALID_ - PACKET</i>	The compressed data passed is corrupted or of an unsupported type
-----------------------------------	---

3.2.2.12 `int opus_packet_get_samples_per_frame ( const unsigned char * data, opus_int32 Fs )`

Gets the number of samples per frame from an Opus packet.

#### Parameters

in	<i>data</i>	char*: Opus packet
in	<i>Fs</i>	opus_int32: Sampling rate in Hz

#### Returns

Number of samples per frame

#### Return values

<i>OPUS_INVALID_PACKET</i>	The compressed data passed is corrupted or of an unsupported type
----------------------------	---

3.2.2.13 `int opus_packet_parse ( const unsigned char * data, int len, unsigned char * out_toc, const unsigned char * frames[48], short size[48], int * payload_offset )`

Parse an opus packet into one or more frames.

Opus\_decode will perform this operation internally so most applications do not need to use this function. This function does not copy the frames, the returned pointers are pointers into the input packet.

#### Parameters

in	<i>data</i>	char*: Opus packet to be parsed
in	<i>len</i>	int: size of data
out	<i>out_toc</i>	char*: TOC pointer
out	<i>frames</i>	char*[48] encapsulated frames
out	<i>size</i>	short[48] sizes of the encapsulated frames
out	<i>payload_offset</i>	int*: returns the position of the payload within the packet (in bytes)

#### Returns

number of frames

## 3.3 Repackitizer

#### Typedefs

- typedef struct [OpusRepackitizer](#) [OpusRepackitizer](#)

## Functions

- int `opus_repacketizer_get_size` (void)
- `OpusRepacketizer *` `opus_repacketizer_init` (`OpusRepacketizer *`rp)
- `OpusRepacketizer *` `opus_repacketizer_create` (void)
- void `opus_repacketizer_destroy` (`OpusRepacketizer *`rp)
- int `opus_repacketizer_cat` (`OpusRepacketizer *`rp, const unsigned char \*data, int len)
- int `opus_repacketizer_out_range` (`OpusRepacketizer *`rp, int begin, int end, unsigned char \*data, int maxlen)
- int `opus_repacketizer_get_nb_frames` (`OpusRepacketizer *`rp)
- int `opus_repacketizer_out` (`OpusRepacketizer *`rp, unsigned char \*data, int maxlen)

### 3.3.1 Typedef Documentation

3.3.1.1 typedef struct `OpusRepacketizer` `OpusRepacketizer`

### 3.3.2 Function Documentation

3.3.2.1 int `opus_repacketizer_cat` ( `OpusRepacketizer *`rp, const unsigned char \*data, int len )

3.3.2.2 `OpusRepacketizer*` `opus_repacketizer_create` ( void )

3.3.2.3 void `opus_repacketizer_destroy` ( `OpusRepacketizer *`rp )

3.3.2.4 int `opus_repacketizer_get_nb_frames` ( `OpusRepacketizer *`rp )

3.3.2.5 int `opus_repacketizer_get_size` ( void )

3.3.2.6 `OpusRepacketizer*` `opus_repacketizer_init` ( `OpusRepacketizer *`rp )

3.3.2.7 int `opus_repacketizer_out` ( `OpusRepacketizer *`rp, unsigned char \*data, int maxlen )

3.3.2.8 int `opus_repacketizer_out_range` ( `OpusRepacketizer *`rp, int begin, int end, unsigned char \*data, int maxlen )

## 3.4 Error codes

### Defines

- #define `OPUS_OK`  
No error.
- #define `OPUS_BAD_ARG`  
One or more invalid/out of range arguments.

- `#define OPUS_BUFFER_TOO_SMALL`  
*The mode struct passed is invalid.*
- `#define OPUS_INTERNAL_ERROR`  
*An internal error was detected.*
- `#define OPUS_INVALID_PACKET`  
*The compressed data passed is corrupted.*
- `#define OPUS_UNIMPLEMENTED`  
*Invalid/unsupported request number.*
- `#define OPUS_INVALID_STATE`  
*An encoder or decoder structure is invalid or already freed.*
- `#define OPUS_ALLOC_FAIL`  
*Memory allocation has failed.*

### 3.4.1 Define Documentation

#### 3.4.1.1 `#define OPUS_ALLOC_FAIL`

Memory allocation has failed.

#### 3.4.1.2 `#define OPUS_BAD_ARG`

One or more invalid/out of range arguments.

#### 3.4.1.3 `#define OPUS_BUFFER_TOO_SMALL`

The mode struct passed is invalid.

#### 3.4.1.4 `#define OPUS_INTERNAL_ERROR`

An internal error was detected.

#### 3.4.1.5 `#define OPUS_INVALID_PACKET`

The compressed data passed is corrupted.

#### 3.4.1.6 `#define OPUS_INVALID_STATE`

An encoder or decoder structure is invalid or already freed.

#### 3.4.1.7 `#define OPUS_OK`

No error.



## 3.4.1.8 #define OPUS\_UNIMPLEMENTED

Invalid/unsupported request number.

## 3.5 Encoder related CTLs

### Defines

- #define **OPUS\_SET\_COMPLEXITY**(x)  
*Configures the encoder's computational complexity.*
- #define **OPUS\_GET\_COMPLEXITY**(x)  
*Gets the encoder's complexity configuration,.*
- #define **OPUS\_SET\_BITRATE**(x)  
*Configures the bitrate in the encoder.*
- #define **OPUS\_GET\_BITRATE**(x)  
*Gets the encoder's bitrate configuration,.*
- #define **OPUS\_SET\_VBR**(x)  
*Configures VBR in the encoder.*
- #define **OPUS\_GET\_VBR**(x)  
*Gets the encoder's VBR configuration,.*
- #define **OPUS\_SET\_VBR\_CONSTRAINT**(x)  
*Configures constrained VBR in the encoder.*
- #define **OPUS\_GET\_VBR\_CONSTRAINT**(x)  
*Gets the encoder's constrained VBR configuration.*
- #define **OPUS\_SET\_FORCE\_CHANNELS**(x)  
*Configures mono/stereo forcing in the encoder.*
- #define **OPUS\_GET\_FORCE\_CHANNELS**(x)  
*Gets the encoder's forced channel configuration,.*
- #define **OPUS\_SET\_BANDWIDTH**(x)  
*Configures the encoder's bandpass,.*
- #define **OPUS\_SET\_SIGNAL**(x)  
*Configures the type of signal being encoded.*
- #define **OPUS\_GET\_SIGNAL**(x)  
*Gets the encoder's configured signal type,.*
- #define **OPUS\_SET\_VOICE\_RATIO**(x)  
*Configures the encoder's expected percentage of voice opposed to music or other signals.*
- #define **OPUS\_GET\_VOICE\_RATIO**(x)  
*Gets the encoder's configured voice ratio value,.*
- #define **OPUS\_SET\_APPLICATION**(x)  
*Configures the encoder's intended application.*
- #define **OPUS\_GET\_APPLICATION**(x)  
*Gets the encoder's configured application,.*

- `#define OPUS_SET_RESTRICTED_LOWDELAY(x)`  
*Configures low-delay mode that disables the speech-optimized mode in exchange for slightly reduced delay.*
- `#define OPUS_GET_RESTRICTED_LOWDELAY(x)`  
*Gets the encoder's forced channel configuration,.*
- `#define OPUS_GET_LOOKAHEAD(x)`  
*Gets the total samples of delay added by the entire codec.*
- `#define OPUS_SET_INBAND_FEC(x)`  
*Configures the encoder's use of inband forward error correction.*
- `#define OPUS_GET_INBAND_FEC(x)`  
*Gets encoder's configured use of inband forward error correction,.*
- `#define OPUS_SET_PACKET_LOSS_PERC(x)`  
*Configures the encoder's expected packet loss percentage.*
- `#define OPUS_GET_PACKET_LOSS_PERC(x)`  
*Gets the encoder's configured packet loss percentage,.*
- `#define OPUS_SET_DTX(x)`  
*Configures the encoder's use of discontinuous transmission.*
- `#define OPUS_GET_DTX(x)`  
*Gets encoder's configured use of discontinuous transmission,.*

### 3.5.1 Detailed Description

#### See also

[Generic CTLs](#), [Opus Encoder](#)

### 3.5.2 Define Documentation

#### 3.5.2.1 `#define OPUS_GET_APPLICATION( x )`

Gets the encoder's configured application,.

#### See also

[OPUS\\_SET\\_APPLICATION](#)

#### Parameters

out	x	int*: Application value
-----	---	-------------------------

#### 3.5.2.2 `#define OPUS_GET_BITRATE( x )`

Gets the encoder's bitrate configuration,.

See also

[OPUS\\_SET\\_BITRATE](#)

Parameters

out	x	opus_int32*: bitrate in bits per second.
-----	---	--

3.5.2.3 `#define OPUS_GET_COMPLEXITY( x )`

Gets the encoder's complexity configuration,.

See also

[OPUS\\_SET\\_COMPLEXITY](#)

Parameters

out	x	int*: 0-10, inclusive
-----	---	-----------------------

3.5.2.4 `#define OPUS_GET_DTX( x )`

Gets encoder's configured use of discontinuous transmission,.

See also

[OPUS\\_SET\\_DTX](#)

Parameters

out	x	int*: DTX flag
-----	---	----------------

3.5.2.5 `#define OPUS_GET_FORCE_CHANNELS( x )`

Gets the encoder's forced channel configuration,.

See also

[OPUS\\_SET\\_FORCE\\_CHANNELS](#)

Parameters

out	x	int*: OPUS_AUTO; 0; 1
-----	---	-----------------------

3.5.2.6 `#define OPUS_GET_INBAND_FEC( x )`

Gets encoder's configured use of inband forward error correction,.

**See also**

[OPUS\\_SET\\_INBAND\\_FEC](#)

**Parameters**

out	x	int*: FEC flag
-----	---	----------------

**3.5.2.7 #define OPUS\_GET\_LOOKAHEAD( x )**

Gets the total samples of delay added by the entire codec.

This can be queried by the encoder and then the provided number of samples can be skipped on from the start of the decoder's output to provide time aligned input and output. From the perspective of a decoding application the real data begins this many samples late.

The decoder contribution to this delay is identical for all decoders, but the encoder portion of the delay may vary from implementation to implementation, version to version, or even depend on the encoder's initial configuration. Applications needing delay compensation should call this CTL rather than hard-coding a value.

**Parameters**

out	x	int*: Number of lookahead samples
-----	---	-----------------------------------

**3.5.2.8 #define OPUS\_GET\_PACKET\_LOSS\_PERC( x )**

Gets the encoder's configured packet loss percentage,.

**See also**

[OPUS\\_SET\\_PACKET\\_LOSS\\_PERC](#)

**Parameters**

out	x	int*: Loss percentage in the range 0-100, inclusive.
-----	---	--

**3.5.2.9 #define OPUS\_GET\_RESTRICTED\_LOWDELAY( x )**

Gets the encoder's forced channel configuration,.

**See also**

[OPUS\\_SET\\_RESTRICTED\\_LOWDELAY](#)

**Parameters**

out	x	int*: 0; 1
-----	---	------------

#### 3.5.2.10 #define OPUS\_GET\_SIGNAL( x )

Gets the encoder's configured signal type,.

**See also**

[OPUS\\_SET\\_SIGNAL](#)

##### Parameters

out	x	int*: Signal type
-----	---	-------------------

#### 3.5.2.11 #define OPUS\_GET\_VBR( x )

Gets the encoder's VBR configuration,.

**See also**

[OPUS\\_SET\\_VBR](#)

##### Parameters

out	x	int*: 0; 1
-----	---	------------

#### 3.5.2.12 #define OPUS\_GET\_VBR\_CONSTRAINT( x )

Gets the encoder's constrained VBR configuration.

**See also**

[OPUS\\_SET\\_VBR\\_CONSTRAINT](#)

##### Parameters

out	x	int*: 0; 1
-----	---	------------

#### 3.5.2.13 #define OPUS\_GET\_VOICE\_RATIO( x )

Gets the encoder's configured voice ratio value,.

**See also**

[OPUS\\_SET\\_VOICE\\_RATIO](#)

##### Parameters

out	x	int*: Voice percentage in the range 0-100, inclusive.
-----	---	---

### 3.5.2.14 #define OPUS\_SET\_APPLICATION( x )

Configures the encoder's intended application.

The initial value is a mandatory argument to the encoder\_create function. The supported values are:

- OPUS\_APPLICATION\_VOIP Process signal for improved speech intelligibility
- OPUS\_APPLICATION\_AUDIO Favor faithfulness to the original input

#### Parameters

in	x	int: Application value
----	---	------------------------

### 3.5.2.15 #define OPUS\_SET\_BANDWIDTH( x )

Configures the encoder's bandpass,.

#### See also

[OPUS\\_GET\\_BANDWIDTH](#) The supported values are:

- OPUS\_BANDWIDTH\_AUTO (default)
- OPUS\_BANDWIDTH\_NARROWBAND 4kHz passband
- OPUS\_BANDWIDTH\_MEDIUMBAND 6kHz passband
- OPUS\_BANDWIDTH\_WIDEBAND 8kHz passband
- OPUS\_BANDWIDTH\_SUPERWIDEBAND 12kHz passband
- OPUS\_BANDWIDTH\_FULLBAND 20kHz passband

#### Parameters

in	x	int: Bandwidth value
----	---	----------------------

### 3.5.2.16 #define OPUS\_SET\_BITRATE( x )

Configures the bitrate in the encoder.

Rates from 500 to 512000 bits per second are meaningful as well as the special values OPUS\_BITRATE\_AUTO and OPUS\_BITRATE\_MAX. The value OPUS\_BITRATE\_MAX can be used to cause the codec to use as much rate as it can, which is useful for controlling the rate by adjusting the output buffer size.

#### Parameters

in	x	opus_int32: bitrate in bits per second.
----	---	---

**3.5.2.17 #define OPUS\_SET\_COMPLEXITY( x )**

Configures the encoder's computational complexity.

The supported range is 0-10 inclusive with 10 representing the highest complexity. The default value is 10.

**Parameters**

in	x	int: 0-10, inclusive
----	---	----------------------

**3.5.2.18 #define OPUS\_SET\_DTX( x )**

Configures the encoder's use of discontinuous transmission.

**Note**

This is only applicable to the LPC layer

**Parameters**

in	x	int: DTX flag, 0 (disabled) is default
----	---	--

**3.5.2.19 #define OPUS\_SET\_FORCE\_CHANNELS( x )**

Configures mono/stereo forcing in the encoder.

This is useful when the caller knows that the input signal is currently a mono source embedded in a stereo stream.

**Parameters**

in	x	int: OPUS_AUTO (default); 1 (forced mono); 2 (forced stereo)
----	---	--

**3.5.2.20 #define OPUS\_SET\_INBAND\_FEC( x )**

Configures the encoder's use of inband forward error correction.

**Note**

This is only applicable to the LPC layer

**Parameters**

in	x	int: FEC flag, 0 (disabled) is default
----	---	--

**3.5.2.21 #define OPUS\_SET\_PACKET\_LOSS\_PERC( x )**

Configures the encoder's expected packet loss percentage.

Higher values will trigger progressively more loss resistant behavior in the encoder at the expense of quality at a given bitrate in the lossless case, but greater quality under loss.

**Parameters**

in	x	int: Loss percentage in the range 0-100, inclusive.
----	---	---

**3.5.2.22 #define OPUS\_SET\_RESTRICTED\_LOWDELAY( x )**

Configures low-delay mode that disables the speech-optimized mode in exchange for slightly reduced delay.

This is useful when the caller knows that the speech-optimized modes will not be needed (use with caution). The setting can only be changed right after initialization or after a reset and changes the lookahead.

**Parameters**

in	x	int: 0 (default); 1 (lowdelay)
----	---	--------------------------------

**3.5.2.23 #define OPUS\_SET\_SIGNAL( x )**

Configures the type of signal being encoded.

This is a hint which helps the encoder's mode selection. The supported values are:

- OPUS\_SIGNAL\_AUTO (default)
- OPUS\_SIGNAL\_VOICE
- OPUS\_SIGNAL\_MUSIC

**Parameters**

in	x	int: Signal type
----	---	------------------

**3.5.2.24 #define OPUS\_SET\_VBR( x )**

Configures VBR in the encoder.

The following values are currently supported:

- 0 CBR
- 1 VBR (default) The configured bitrate may not be met exactly because frames must be an integer number of bytes in length.



**Warning**

Only the MDCT mode of Opus can provide hard CBR behavior.

**Parameters**

in	x	int: 0; 1 (default)
----	---	---------------------

**3.5.2.25 #define OPUS\_SET\_VBR\_CONSTRAINT( x )**

Configures constrained VBR in the encoder.

The following values are currently supported:

- 0 Unconstrained VBR (default)
- 1 Maximum one frame buffering delay assuming transport with a serialization speed of the nominal bitrate This setting is irrelevant when the encoder is in CBR mode.

**Warning**

Only the MDCT mode of Opus currently heeds the constraint. Speech mode ignores it completely, hybrid mode may fail to obey it if the LPC layer uses more bitrate than the constraint would have permitted.

**Parameters**

in	x	int: 0 (default); 1
----	---	---------------------

**3.5.2.26 #define OPUS\_SET\_VOICE\_RATIO( x )**

Configures the encoder's expected percentage of voice opposed to music or other signals.

**Note**

This interface is currently more aspiration than actuality. It's ultimately expected to bias an automatic signal classifier, but it currently just shifts the static bitrate to mode mapping around a little bit.

**Parameters**

in	x	int: Voice percentage in the range 0-100, inclusive.
----	---	--

**3.6 Generic CTLs****Defines**

- #define [OPUS\\_RESET\\_STATE](#)

*Resets the codec state to be equivalent to a freshly initialized state.*

- `#define OPUS_GET_FINAL_RANGE(x)`

*Gets the final state of the codec's entropy coder.*

- `#define OPUS_GET_PITCH(x)`

*Gets the pitch of the last decoded frame, if available.*

- `#define OPUS_GET_BANDWIDTH(x)`

*Gets the encoder's configured bandpass or the decoder's last bandpass,.*

### 3.6.1 Detailed Description

#### See also

[opus\\_encoder\\_ctl](#), [Opus Encoder](#), [Opus Decoder](#)

### 3.6.2 Define Documentation

#### 3.6.2.1 `#define OPUS_GET_BANDWIDTH( x )`

Gets the encoder's configured bandpass or the decoder's last bandpass,.

#### See also

[OPUS\\_SET\\_BANDWIDTH](#)

#### Parameters

out	x	int*: Bandwidth value
-----	---	-----------------------

#### 3.6.2.2 `#define OPUS_GET_FINAL_RANGE( x )`

Gets the final state of the codec's entropy coder.

This is used for testing purposes, The encoder and decoder state should be identical after coding a payload (assuming no data corruption or software bugs)

#### Parameters

out	x	opus_uint32*: Entropy coder state
-----	---	-----------------------------------

#### 3.6.2.3 `#define OPUS_GET_PITCH( x )`

Gets the pitch of the last decoded frame, if available.

This can be used for any post-processing algorithm requiring the use of pitch, e.g. time stretching/shortening. If the last frame was not voiced, or if the pitch was not coded in the frame, then zero is returned.

**Parameters**

out	x	opus_int32*: pitch period at 48 kHz (or 0 if not available)
-----	---	---

**3.6.2.4 #define OPUS\_RESET\_STATE**

Resets the codec state to be equivalent to a freshly initialized state.

This should be called when switching streams in order to prevent the back to back decoding from giving different results from one at a time decoding.

**3.7 Opus library information functions****Functions**

- const char \* [opus\\_strerror](#) (int error)  
*Converts an opus error code into a human readable string.*
- const char \* [opus\\_get\\_version\\_string](#) (void)  
*Gets the libopus version string.*

**3.7.1 Function Documentation****3.7.1.1 const char\* opus\_get\_version\_string ( void )**

Gets the libopus version string.

**Returns**

Version string

**3.7.1.2 const char\* opus\_strerror ( int error )**

Converts an opus error code into a human readable string.

**Parameters**

in	error	int: Error number
----	-------	-------------------

**Returns**

Error string



## Chapter 4

# File Documentation

### 4.1 opus.h File Reference

Opus reference implementation API.

```
#include "opus_types.h"
#include "opus_defines.h"
```

#### Typedefs

- typedef struct [OpusEncoder](#) [OpusEncoder](#)  
*Opus encoder state.*
- typedef struct [OpusDecoder](#) [OpusDecoder](#)  
*Opus decoder state.*
- typedef struct [OpusRepacketizer](#) [OpusRepacketizer](#)

#### Functions

- int [opus\\_encoder\\_get\\_size](#) (int channels)
- [OpusEncoder](#) \* [opus\\_encoder\\_create](#) ([opus\\_int32](#) Fs, int channels, int application, int \*error)  
*Allocates and initializes an encoder state.*
- int [opus\\_encoder\\_init](#) ([OpusEncoder](#) \*st, [opus\\_int32](#) Fs, int channels, int application)  
*Initializes a previously allocated encoder state The memory pointed to by st must be the size returned by opus\_encoder\_get\_size.*
- int [opus\\_encode](#) ([OpusEncoder](#) \*st, const [opus\\_int16](#) \*pcm, int frame\_size, unsigned char \*data, int max\_data\_bytes)  
*Encodes an Opus frame.*
- int [opus\\_encode\\_float](#) ([OpusEncoder](#) \*st, const float \*pcm, int frame\_size, unsigned char \*data, int max\_data\_bytes)

- Encodes an Opus frame from floating point input.*

  - void `opus_encoder_destroy` (`OpusEncoder` \*st)

*Frees an OpusEncoder allocated by opus\_encoder\_create.*
- int `opus_encoder_ctl` (`OpusEncoder` \*st, int request,...)

*Perform a CTL function on an Opus encoder.*
- int `opus_decoder_get_size` (int channels)

*Gets the size of an OpusDecoder structure.*
- `OpusDecoder` \* `opus_decoder_create` (`opus_int32` Fs, int channels, int \*error)

*Allocates and initializes a decoder state.*
- int `opus_decoder_init` (`OpusDecoder` \*st, `opus_int32` Fs, int channels)

*Initializes a previously allocated decoder state.*
- int `opus_decode` (`OpusDecoder` \*st, const unsigned char \*data, int len, `opus_int16` \*pcm, int frame\_size, int decode\_fec)

*Decode an Opus frame.*
- int `opus_decode_float` (`OpusDecoder` \*st, const unsigned char \*data, int len, float \*pcm, int frame\_size, int decode\_fec)

*Decode an opus frame with floating point output.*
- int `opus_decoder_ctl` (`OpusDecoder` \*st, int request,...)

*Perform a CTL function on an Opus decoder.*
- void `opus_decoder_destroy` (`OpusDecoder` \*st)

*Frees an OpusDecoder allocated by opus\_decoder\_create.*
- int `opus_packet_parse` (const unsigned char \*data, int len, unsigned char \*out\_toc, const unsigned char \*frames[48], short size[48], int \*payload\_offset)

*Parse an opus packet into one or more frames.*
- int `opus_packet_get_bandwidth` (const unsigned char \*data)

*Gets the bandwidth of an Opus packet.*
- int `opus_packet_get_samples_per_frame` (const unsigned char \*data, `opus_int32` Fs)

*Gets the number of samples per frame from an Opus packet.*
- int `opus_packet_get_nb_channels` (const unsigned char \*data)

*Gets the number of channels from an Opus packet.*
- int `opus_packet_get_nb_frames` (const unsigned char packet[], int len)

*Gets the number of frame in an Opus packet.*
- int `opus_decoder_get_nb_samples` (const `OpusDecoder` \*dec, const unsigned char packet[], int len)

*Gets the number of samples of an Opus packet.*
- int `opus_repacketizer_get_size` (void)
- `OpusRepacketizer` \* `opus_repacketizer_init` (`OpusRepacketizer` \*rp)
- `OpusRepacketizer` \* `opus_repacketizer_create` (void)
- void `opus_repacketizer_destroy` (`OpusRepacketizer` \*rp)
- int `opus_repacketizer_cat` (`OpusRepacketizer` \*rp, const unsigned char \*data, int len)
- int `opus_repacketizer_out_range` (`OpusRepacketizer` \*rp, int begin, int end, unsigned char \*data, int maxlen)
- int `opus_repacketizer_get_nb_frames` (`OpusRepacketizer` \*rp)
- int `opus_repacketizer_out` (`OpusRepacketizer` \*rp, unsigned char \*data, int maxlen)

### 4.1.1 Detailed Description

Opus reference implementation API.

## 4.2 opus\_defines.h File Reference

Opus reference implementation constants.

```
#include "opus_types.h"
```

### Defines

- #define [OPUS\\_OK](#)  
*No error.*
- #define [OPUS\\_BAD\\_ARG](#)  
*One or more invalid/out of range arguments.*
- #define [OPUS\\_BUFFER\\_TOO\\_SMALL](#)  
*The mode struct passed is invalid.*
- #define [OPUS\\_INTERNAL\\_ERROR](#)  
*An internal error was detected.*
- #define [OPUS\\_INVALID\\_PACKET](#)  
*The compressed data passed is corrupted.*
- #define [OPUS\\_UNIMPLEMENTED](#)  
*Invalid/unsupported request number.*
- #define [OPUS\\_INVALID\\_STATE](#)  
*An encoder or decoder structure is invalid or already freed.*
- #define [OPUS\\_ALLOC\\_FAIL](#)  
*Memory allocation has failed.*
- #define [OPUS\\_SET\\_COMPLEXITY\(x\)](#)  
*Configures the encoder's computational complexity.*
- #define [OPUS\\_GET\\_COMPLEXITY\(x\)](#)  
*Gets the encoder's complexity configuration,.*
- #define [OPUS\\_SET\\_BITRATE\(x\)](#)  
*Configures the bitrate in the encoder.*
- #define [OPUS\\_GET\\_BITRATE\(x\)](#)  
*Gets the encoder's bitrate configuration,.*
- #define [OPUS\\_SET\\_VBR\(x\)](#)  
*Configures VBR in the encoder.*
- #define [OPUS\\_GET\\_VBR\(x\)](#)  
*Gets the encoder's VBR configuration,.*
- #define [OPUS\\_SET\\_VBR\\_CONSTRAINT\(x\)](#)  
*Configures constrained VBR in the encoder.*
- #define [OPUS\\_GET\\_VBR\\_CONSTRAINT\(x\)](#)

- Gets the encoder's constrained VBR configuration.*

  - #define [OPUS\\_SET\\_FORCE\\_CHANNELS\(x\)](#)

*Configures mono/stereo forcing in the encoder.*

- #define [OPUS\\_GET\\_FORCE\\_CHANNELS\(x\)](#)

*Gets the encoder's forced channel configuration,.*

- #define [OPUS\\_SET\\_BANDWIDTH\(x\)](#)

*Configures the encoder's bandpass,.*

- #define [OPUS\\_SET\\_SIGNAL\(x\)](#)

*Configures the type of signal being encoded.*

- #define [OPUS\\_GET\\_SIGNAL\(x\)](#)

*Gets the encoder's configured signal type,.*

- #define [OPUS\\_SET\\_VOICE\\_RATIO\(x\)](#)

*Configures the encoder's expected percentage of voice opposed to music or other signals.*

- #define [OPUS\\_GET\\_VOICE\\_RATIO\(x\)](#)

*Gets the encoder's configured voice ratio value,.*

- #define [OPUS\\_SET\\_APPLICATION\(x\)](#)

*Configures the encoder's intended application.*

- #define [OPUS\\_GET\\_APPLICATION\(x\)](#)

*Gets the encoder's configured application,.*

- #define [OPUS\\_SET\\_RESTRICTED\\_LOWDELAY\(x\)](#)

*Configures low-delay mode that disables the speech-optimized mode in exchange for slightly reduced delay.*

- #define [OPUS\\_GET\\_RESTRICTED\\_LOWDELAY\(x\)](#)

*Gets the encoder's forced channel configuration,.*

- #define [OPUS\\_GET\\_LOOKAHEAD\(x\)](#)

*Gets the total samples of delay added by the entire codec.*

- #define [OPUS\\_SET\\_INBAND\\_FEC\(x\)](#)

*Configures the encoder's use of inband forward error correction.*

- #define [OPUS\\_GET\\_INBAND\\_FEC\(x\)](#)

*Gets encoder's configured use of inband forward error correction,.*

- #define [OPUS\\_SET\\_PACKET\\_LOSS\\_PERC\(x\)](#)

*Configures the encoder's expected packet loss percentage.*

- #define [OPUS\\_GET\\_PACKET\\_LOSS\\_PERC\(x\)](#)

*Gets the encoder's configured packet loss percentage,.*

- #define [OPUS\\_SET\\_DTX\(x\)](#)

*Configures the encoder's use of discontinuous transmission.*

- #define [OPUS\\_GET\\_DTX\(x\)](#)

*Gets encoder's configured use of discontinuous transmission,.*

- #define [OPUS\\_RESET\\_STATE](#)

*Resets the codec state to be equivalent to a freshly initialized state.*

- #define [OPUS\\_GET\\_FINAL\\_RANGE\(x\)](#)

*Gets the final state of the codec's entropy coder.*

- #define [OPUS\\_GET\\_PITCH\(x\)](#)



*Gets the pitch of the last decoded frame, if available.*

- #define [OPUS\\_GET\\_BANDWIDTH\(x\)](#)

*Gets the encoder's configured bandpass or the decoder's last bandpass,.*

## Functions

- const char \* [opus\\_strerror](#) (int error)

*Converts an opus error code into a human readable string.*

- const char \* [opus\\_get\\_version\\_string](#) (void)

*Gets the libopus version string.*

### 4.2.1 Detailed Description

Opus reference implementation constants.

## 4.3 opus\_multistream.h File Reference

Opus reference implementation multistream API.

```
#include "opus.h"
```

## Defines

- #define [\\_\\_opus\\_check\\_encstate\\_ptr\(ptr\)](#) ((ptr) + ((ptr) - ([OpusEncoder\\*\\*](#))(ptr)))
- #define [\\_\\_opus\\_check\\_decstate\\_ptr\(ptr\)](#) ((ptr) + ((ptr) - ([OpusDecoder\\*\\*](#))(ptr)))
- #define [OPUS\\_MULTISTREAM\\_GET\\_ENCODER\\_STATE\\_REQUEST](#) 5120
- #define [OPUS\\_MULTISTREAM\\_GET\\_DECODER\\_STATE\\_REQUEST](#) 5122
- #define [OPUS\\_MULTISTREAM\\_GET\\_ENCODER\\_STATE](#)(x, y) [OPUS\\_MULTISTREAM\\_GET\\_ENCODER\\_STATE\\_REQUEST](#), [\\_\\_opus\\_check\\_int](#)(x), [\\_\\_opus\\_check\\_encstate\\_ptr](#)(y)
- #define [OPUS\\_MULTISTREAM\\_GET\\_DECODER\\_STATE](#)(x, y) [OPUS\\_MULTISTREAM\\_GET\\_DECODER\\_STATE\\_REQUEST](#), [\\_\\_opus\\_check\\_int](#)(x), [\\_\\_opus\\_check\\_decstate\\_ptr](#)(y)

## Typedefs

- typedef struct [OpusMSEncoder](#) [OpusMSEncoder](#)
- typedef struct [OpusMSDecoder](#) [OpusMSDecoder](#)

## Functions

- `OpusMSEncoder * opus_multistream_encoder_create (opus_int32 Fs, int channels, int streams, int coupled_streams, unsigned char *mapping, int application, int *error)`

*Allocate and initialize a multistream encoder state object.*

- `int opus_multistream_encoder_init (OpusMSEncoder *st, opus_int32 Fs, int channels, int streams, int coupled_streams, unsigned char *mapping, int application)`

*Initialize an already allocated multistream encoder state.*

- `int opus_multistream_encode (OpusMSEncoder *st, const opus_int16 *pcm, int frame_size, unsigned char *data, int max_data_bytes)`

*Returns length of the data payload (in bytes) or a negative error code.*

- `int opus_multistream_encode_float (OpusMSEncoder *st, const float *pcm, int frame_size, unsigned char *data, int max_data_bytes)`

*Returns length of the data payload (in bytes) or a negative error code.*

- `void opus_multistream_encoder_destroy (OpusMSEncoder *st)`

*Deallocate a multistream encoder state.*

- `int opus_multistream_encoder_ctl (OpusMSEncoder *st, int request,...)`

*Get or set options on a multistream encoder state.*

- `OpusMSDecoder * opus_multistream_decoder_create (opus_int32 Fs, int channels, int streams, int coupled_streams, unsigned char *mapping, int *error)`

*Allocate and initialize a multistream decoder state object.*

- `int opus_multistream_decoder_init (OpusMSDecoder *st, opus_int32 Fs, int channels, int streams, int coupled_streams, unsigned char *mapping)`

*Initialize a previously allocated decoder state object.*

- `int opus_multistream_decode (OpusMSDecoder *st, const unsigned char *data, int len, opus_int16 *pcm, int frame_size, int decode_fec)`

*Returns the number of samples decoded or a negative error code.*

- `int opus_multistream_decode_float (OpusMSDecoder *st, const unsigned char *data, int len, float *pcm, int frame_size, int decode_fec)`

*Returns the number of samples decoded or a negative error code.*

- `int opus_multistream_decoder_ctl (OpusMSDecoder *st, int request,...)`

*Get or set options on a multistream decoder state.*

- `void opus_multistream_decoder_destroy (OpusMSDecoder *st)`

*Deallocate a multistream decoder state object.*

### 4.3.1 Detailed Description

Opus reference implementation multistream API.

### 4.3.2 Define Documentation

4.3.2.1 `#define __opus_check_decstate_ptr( ptr ) ((ptr) + ((ptr) - (OpusDecoder**)(ptr)))`

4.3.2.2 `#define __opus_check_encstate_ptr( ptr ) ((ptr) + ((ptr) - (OpusEncoder**)(ptr)))`

4.3.2.3 `#define OPUS_MULTISTREAM_GET_DECODER_STATE( x, y  
 ) OPUS_MULTISTREAM_GET_DECODER_STATE_REQUEST, __opus_check_int(x),  
 __opus_check_decstate_ptr(y)`

4.3.2.4 `#define OPUS_MULTISTREAM_GET_DECODER_STATE_REQUEST 5122`

4.3.2.5 `#define OPUS_MULTISTREAM_GET_ENCODER_STATE( x, y  
 ) OPUS_MULTISTREAM_GET_ENCODER_STATE_REQUEST, __opus_check_int(x),  
 __opus_check_encstate_ptr(y)`

4.3.2.6 `#define OPUS_MULTISTREAM_GET_ENCODER_STATE_REQUEST 5120`

### 4.3.3 Typedef Documentation

4.3.3.1 `typedef struct OpusMSDecoder OpusMSDecoder`

4.3.3.2 `typedef struct OpusMSEncoder OpusMSEncoder`

### 4.3.4 Function Documentation

4.3.4.1 `int opus_multistream_decode ( OpusMSDecoder * st, const unsigned char * data,  
 int len, opus_int16 * pcm, int frame_size, int decode_fec )`

Returns the number of samples decoded or a negative error code.

#### Parameters

<i>st</i>	Decoder state
<i>data</i>	Input payload. Use a NULL pointer to indicate packet loss
<i>len</i>	Number of bytes in payload
<i>pcm</i>	Output signal, samples interleaved in channel order . length is frame_size*channels
<i>frame_size</i>	Number of samples per frame of input signal
<i>decode_fec</i>	Flag (0/1) to request that any in-band forward error correction data be decoded. If no such data is available the frame is decoded as if it were lost.

4.3.4.2 `int opus_multistream_decode_float ( OpusMSDecoder * st, const unsigned char *  
 data, int len, float * pcm, int frame_size, int decode_fec )`

Returns the number of samples decoded or a negative error code.

**Parameters**

<i>st</i>	Decoder state
<i>data</i>	Input payload buffer. Use a NULL pointer to indicate packet loss
<i>len</i>	Number of payload bytes in data
<i>pcm</i>	Buffer for the output signal (interleaved iin channel order). length is frame_size*channels
<i>frame_size</i>	Number of samples per frame of input signal
<i>decode_fec</i>	Flag (0/1) to request that any in-band forward error correction data be decoded. If no such data is available the frame is decoded as if it were lost.

#### 4.3.4.3 **OpusMSDecoder\*** opus\_multistream\_decoder\_create ( **opus\_int32** *Fs*, int *channels*, int *streams*, int *coupled\_streams*, unsigned char \* *mapping*, int \* *error* )

Allocate and initialize a multistream decoder state object.

Call [opus\\_multistream\\_decoder\\_destroy\(\)](#) to release this object when finished.

**Parameters**

<i>Fs</i>	Sampling rate to decode at (Hz)
<i>channels</i>	Number of channels to decode
<i>streams</i>	Total number of coded streams in the multistream
<i>coupled_streams</i>	Number of coupled (stereo) streams in the multistream
<i>mapping</i>	Stream to channel mapping table
<i>error</i>	Error code

#### 4.3.4.4 int opus\_multistream\_decoder\_ctl ( **OpusMSDecoder** \* *st*, int *request*, ... )

Get or set options on a multistream decoder state.

#### 4.3.4.5 void opus\_multistream\_decoder\_destroy ( **OpusMSDecoder** \* *st* )

Deallocate a multistream decoder state object.

#### 4.3.4.6 int opus\_multistream\_decoder\_init ( **OpusMSDecoder** \* *st*, **opus\_int32** *Fs*, int *channels*, int *streams*, int *coupled\_streams*, unsigned char \* *mapping* )

Intialize a previously allocated decoder state object.

**Parameters**

<i>st</i>	Encoder state
<i>Fs</i>	Sample rate of input signal (Hz)
<i>channels</i>	Number of channels in the input signal
<i>streams</i>	Total number of coded streams

<i>coupled_streams</i>	Number of coupled (stereo) streams
<i>mapping</i>	Stream to channel mapping table

4.3.4.7 `int opus_multistream_encode ( OpusMSEncoder * st, const opus_int16 * pcm, int frame_size, unsigned char * data, int max_data_bytes )`

Returns length of the data payload (in bytes) or a negative error code.

#### Parameters

<i>st</i>	Encoder state
<i>pcm</i>	Input signal as interleaved samples. Length is frame_size*channels
<i>frame_size</i>	Number of samples per frame of input signal
<i>data</i>	Output buffer for the compressed payload (no more than max_data_bytes long)
<i>max_data_bytes</i>	Allocated memory for payload; don't use for controlling bitrate

4.3.4.8 `int opus_multistream_encode_float ( OpusMSEncoder * st, const float * pcm, int frame_size, unsigned char * data, int max_data_bytes )`

Returns length of the data payload (in bytes) or a negative error code.

#### Parameters

<i>st</i>	Encoder state
<i>pcm</i>	Input signal interleaved in channel order. length is frame_size*channels
<i>frame_size</i>	Number of samples per frame of input signal
<i>data</i>	Output buffer for the compressed payload (no more than max_data_bytes long)
<i>max_data_bytes</i>	Allocated memory for payload; don't use for controlling bitrate

4.3.4.9 `OpusMSEncoder* opus_multistream_encoder_create ( opus_int32 Fs, int channels, int streams, int coupled_streams, unsigned char * mapping, int application, int * error )`

Allocate and initialize a multistream encoder state object.

Call [opus\\_multistream\\_encoder\\_destroy\(\)](#) to release this object when finished.

#### Parameters

<i>Fs</i>	Sampling rate of input signal (Hz)
<i>channels</i>	Number of channels in the input signal

<i>streams</i>	Total number of streams to encode from the input
<i>coupled_ - streams</i>	Number of coupled (stereo) streams to encode
<i>mapping</i>	Encoded mapping between channels and streams
<i>application</i>	Coding mode (OPUS_APPLICATION_VOIP/OPUS_APPLICATION_AUDIO)
<i>error</i>	Error code

4.3.4.10 `int opus_multistream_encoder_ctl ( OpusMSEncoder * st, int request, ... )`

Get or set options on a multistream encoder state.

4.3.4.11 `void opus_multistream_encoder_destroy ( OpusMSEncoder * st )`

Deallocate a multistream encoder state.

4.3.4.12 `int opus_multistream_encoder_init ( OpusMSEncoder * st, opus_int32 Fs, int channels, int streams, int coupled_streams, unsigned char * mapping, int application )`

Initialize an already allocated multistream encoder state.

#### Parameters

<i>st</i>	Encoder state
<i>Fs</i>	Sampling rate of input signal (Hz)
<i>channels</i>	Number of channels in the input signal
<i>streams</i>	Total number of streams to encode from the input
<i>coupled_ - streams</i>	Number of coupled (stereo) streams to encode
<i>mapping</i>	Encoded mapping between channels and streams
<i>application</i>	Coding mode (OPUS_APPLICATION_VOIP/OPUS_APPLICATION_AUDIO)

## 4.4 opus\_types.h File Reference

Opus reference implementation types.

#### Defines

- #define `opus_int` int
- #define `opus_int64` long long
- #define `opus_int8` signed char

- #define [opus\\_uint](#) unsigned int
- #define [opus\\_uint64](#) unsigned long long
- #define [opus\\_uint8](#) unsigned char

## Typedefs

- typedef short [opus\\_int16](#)
- typedef unsigned short [opus\\_uint16](#)
- typedef int [opus\\_int32](#)
- typedef unsigned int [opus\\_uint32](#)

### 4.4.1 Detailed Description

Opus reference implementation types.

### 4.4.2 Define Documentation

4.4.2.1 #define [opus\\_int](#) int

4.4.2.2 #define [opus\\_int64](#) long long

4.4.2.3 #define [opus\\_int8](#) signed char

4.4.2.4 #define [opus\\_uint](#) unsigned int

4.4.2.5 #define [opus\\_uint64](#) unsigned long long

4.4.2.6 #define [opus\\_uint8](#) unsigned char

### 4.4.3 Typedef Documentation

4.4.3.1 typedef short [opus\\_int16](#)

4.4.3.2 typedef int [opus\\_int32](#)

4.4.3.3 typedef unsigned short [opus\\_uint16](#)

4.4.3.4 typedef unsigned int [opus\\_uint32](#)

# Index

- [\\_\\_opus\\_check\\_decstate\\_ptr](#)
  - [opus\\_multistream.h, 35](#)
- [\\_\\_opus\\_check\\_encstate\\_ptr](#)
  - [opus\\_multistream.h, 35](#)
- [Encoder related CTLs, 17](#)
  - [OPUS\\_GET\\_APPLICATION, 18](#)
  - [OPUS\\_GET\\_BITRATE, 18](#)
  - [OPUS\\_GET\\_COMPLEXITY, 19](#)
  - [OPUS\\_GET\\_DTX, 19](#)
  - [OPUS\\_GET\\_FORCE\\_CHANNELS, 19](#)
  - [OPUS\\_GET\\_INBAND\\_FEC, 19](#)
  - [OPUS\\_GET\\_LOOKAHEAD, 20](#)
  - [OPUS\\_GET\\_PACKET\\_LOSS\\_PERC, 20](#)
  - [OPUS\\_GET\\_RESTRICTED\\_LOWDELAY, 20](#)
  - [OPUS\\_GET\\_SIGNAL, 20](#)
  - [OPUS\\_GET\\_VBR, 21](#)
  - [OPUS\\_GET\\_VBR\\_CONSTRAINT, 21](#)
  - [OPUS\\_GET\\_VOICE\\_RATIO, 21](#)
  - [OPUS\\_SET\\_APPLICATION, 21](#)
  - [OPUS\\_SET\\_BANDWIDTH, 22](#)
  - [OPUS\\_SET\\_BITRATE, 22](#)
  - [OPUS\\_SET\\_COMPLEXITY, 22](#)
  - [OPUS\\_SET\\_DTX, 23](#)
  - [OPUS\\_SET\\_FORCE\\_CHANNELS, 23](#)
  - [OPUS\\_SET\\_INBAND\\_FEC, 23](#)
  - [OPUS\\_SET\\_PACKET\\_LOSS\\_PERC, 23](#)
  - [OPUS\\_SET\\_RESTRICTED\\_LOWDELAY, 24](#)
  - [OPUS\\_SET\\_SIGNAL, 24](#)
  - [OPUS\\_SET\\_VBR, 24](#)
  - [OPUS\\_SET\\_VBR\\_CONSTRAINT, 25](#)
  - [OPUS\\_SET\\_VOICE\\_RATIO, 25](#)
- [Error codes, 15](#)
  - [OPUS\\_ALLOC\\_FAIL, 16](#)
  - [OPUS\\_BAD\\_ARG, 16](#)
  - [OPUS\\_BUFFER\\_TOO\\_SMALL, 16](#)
  - [OPUS\\_INTERNAL\\_ERROR, 16](#)
  - [OPUS\\_INVALID\\_PACKET, 16](#)
  - [OPUS\\_INVALID\\_STATE, 16](#)
  - [OPUS\\_OK, 16](#)
  - [OPUS\\_UNIMPLEMENTED, 16](#)
- [Generic CTLs, 25](#)
  - [OPUS\\_GET\\_BANDWIDTH, 26](#)
  - [OPUS\\_GET\\_FINAL\\_RANGE, 26](#)
  - [OPUS\\_GET\\_PITCH, 26](#)
  - [OPUS\\_RESET\\_STATE, 27](#)
- [Opus Decoder, 8](#)
  - [opus\\_decode, 10](#)
  - [opus\\_decode\\_float, 10](#)
  - [opus\\_decoder\\_create, 10](#)
  - [opus\\_decoder\\_ctl, 11](#)
  - [opus\\_decoder\\_destroy, 11](#)
  - [opus\\_decoder\\_get\\_nb\\_samples, 11](#)
  - [opus\\_decoder\\_get\\_size, 11](#)
  - [opus\\_decoder\\_init, 12](#)
  - [opus\\_packet\\_get\\_bandwidth, 12](#)
  - [opus\\_packet\\_get\\_nb\\_channels, 13](#)
  - [opus\\_packet\\_get\\_nb\\_frames, 13](#)
  - [opus\\_packet\\_get\\_samples\\_per\\_frame, 13](#)
  - [opus\\_packet\\_parse, 14](#)
- [OpusDecoder, 9](#)
- [Opus Encoder, 5](#)
  - [opus\\_encode, 6](#)
  - [opus\\_encode\\_float, 6](#)
  - [opus\\_encoder\\_create, 7](#)
  - [opus\\_encoder\\_ctl, 7](#)
  - [opus\\_encoder\\_destroy, 7](#)
  - [opus\\_encoder\\_get\\_size, 8](#)
  - [opus\\_encoder\\_init, 8](#)
- [OpusEncoder, 6](#)
- [Opus library information functions, 27](#)
  - [opus\\_get\\_version\\_string, 27](#)
  - [opus\\_strerror, 27](#)
- [opus.h, 29](#)
- [OPUS\\_ALLOC\\_FAIL](#)



- Error codes, [16](#)
- OPUS\_BAD\_ARG
  - Error codes, [16](#)
- OPUS\_BUFFER\_TOO\_SMALL
  - Error codes, [16](#)
- opus\_decode
  - Opus Decoder, [10](#)
- opus\_decode\_float
  - Opus Decoder, [10](#)
- opus\_decoder\_create
  - Opus Decoder, [10](#)
- opus\_decoder\_ctl
  - Opus Decoder, [11](#)
- opus\_decoder\_destroy
  - Opus Decoder, [11](#)
- opus\_decoder\_get\_nb\_samples
  - Opus Decoder, [11](#)
- opus\_decoder\_get\_size
  - Opus Decoder, [11](#)
- opus\_decoder\_init
  - Opus Decoder, [12](#)
- opus\_defines.h, [31](#)
- opus\_encode
  - Opus Encoder, [6](#)
- opus\_encode\_float
  - Opus Encoder, [6](#)
- opus\_encoder\_create
  - Opus Encoder, [7](#)
- opus\_encoder\_ctl
  - Opus Encoder, [7](#)
- opus\_encoder\_destroy
  - Opus Encoder, [7](#)
- opus\_encoder\_get\_size
  - Opus Encoder, [8](#)
- opus\_encoder\_init
  - Opus Encoder, [8](#)
- OPUS\_GET\_APPLICATION
  - Encoder related CTLs, [18](#)
- OPUS\_GET\_BANDWIDTH
  - Generic CTLs, [26](#)
- OPUS\_GET\_BITRATE
  - Encoder related CTLs, [18](#)
- OPUS\_GET\_COMPLEXITY
  - Encoder related CTLs, [19](#)
- OPUS\_GET\_DTX
  - Encoder related CTLs, [19](#)
- OPUS\_GET\_FINAL\_RANGE
  - Generic CTLs, [26](#)
- OPUS\_GET\_FORCE\_CHANNELS
  - Encoder related CTLs, [19](#)
- OPUS\_GET\_INBAND\_FEC
  - Encoder related CTLs, [19](#)
- OPUS\_GET\_LOOKAHEAD
  - Encoder related CTLs, [20](#)
- OPUS\_GET\_PACKET\_LOSS\_PERC
  - Encoder related CTLs, [20](#)
- OPUS\_GET\_PITCH
  - Generic CTLs, [26](#)
- OPUS\_GET\_RESTRICTED\_LOWDELAY
  - Encoder related CTLs, [20](#)
- OPUS\_GET\_SIGNAL
  - Encoder related CTLs, [20](#)
- OPUS\_GET\_VBR
  - Encoder related CTLs, [21](#)
- OPUS\_GET\_VBR\_CONSTRAINT
  - Encoder related CTLs, [21](#)
- opus\_get\_version\_string
  - Opus library information functions, [27](#)
- OPUS\_GET\_VOICE\_RATIO
  - Encoder related CTLs, [21](#)
- opus\_int
  - opus\_types.h, [39](#)
- opus\_int16
  - opus\_types.h, [39](#)
- opus\_int32
  - opus\_types.h, [39](#)
- opus\_int64
  - opus\_types.h, [39](#)
- opus\_int8
  - opus\_types.h, [39](#)
- OPUS\_INTERNAL\_ERROR
  - Error codes, [16](#)
- OPUS\_INVALID\_PACKET
  - Error codes, [16](#)
- OPUS\_INVALID\_STATE
  - Error codes, [16](#)
- opus\_multistream.h, [33](#)
  - \_\_opus\_check\_decstate\_ptr, [35](#)
  - \_\_opus\_check\_encstate\_ptr, [35](#)
  - opus\_multistream\_decode, [35](#)
  - opus\_multistream\_decode\_float, [35](#)
  - opus\_multistream\_decoder\_create, [36](#)
  - opus\_multistream\_decoder\_ctl, [36](#)
  - opus\_multistream\_decoder\_destroy, [36](#)
  - opus\_multistream\_decoder\_init, [36](#)
  - opus\_multistream\_encode, [37](#)
  - opus\_multistream\_encode\_float, [37](#)
  - opus\_multistream\_encoder\_create, [37](#)
  - opus\_multistream\_encoder\_ctl, [38](#)
  - opus\_multistream\_encoder\_destroy, [38](#)

- opus\_multistream\_encoder\_init, 38
- OPUS\_MULTISTREAM\_GET\_DECODER\_STATE, 35
- OPUS\_MULTISTREAM\_GET\_DECODER\_STATE\_REQUEST, 35
- OPUS\_MULTISTREAM\_GET\_ENCODER\_STATE, 35
- OPUS\_MULTISTREAM\_GET\_ENCODER\_STATE\_REQUEST, 35
- OpusMSDecoder, 35
- OpusMSEncoder, 35
- opus\_multistream\_decode
  - opus\_multistream.h, 35
- opus\_multistream\_decode\_float
  - opus\_multistream.h, 35
- opus\_multistream\_decoder\_create
  - opus\_multistream.h, 36
- opus\_multistream\_decoder\_ctl
  - opus\_multistream.h, 36
- opus\_multistream\_decoder\_destroy
  - opus\_multistream.h, 36
- opus\_multistream\_decoder\_init
  - opus\_multistream.h, 36
- opus\_multistream\_encode
  - opus\_multistream.h, 37
- opus\_multistream\_encode\_float
  - opus\_multistream.h, 37
- opus\_multistream\_encoder\_create
  - opus\_multistream.h, 37
- opus\_multistream\_encoder\_ctl
  - opus\_multistream.h, 38
- opus\_multistream\_encoder\_destroy
  - opus\_multistream.h, 38
- opus\_multistream\_encoder\_init
  - opus\_multistream.h, 38
- OPUS\_MULTISTREAM\_GET\_DECODER\_STATE
  - opus\_multistream.h, 35
- OPUS\_MULTISTREAM\_GET\_DECODER\_STATE\_REQUEST
  - opus\_multistream.h, 35
- OPUS\_MULTISTREAM\_GET\_ENCODER\_STATE
  - opus\_multistream.h, 35
- OPUS\_MULTISTREAM\_GET\_ENCODER\_STATE\_REQUEST
  - opus\_multistream.h, 35
- OPUS\_OK
  - Error codes, 16
- opus\_packet\_get\_bandwidth
  - Opus Decoder, 12
- opus\_packet\_get\_nb\_channels
  - Opus Decoder, 13
- opus\_packet\_get\_nb\_frames
  - Opus Decoder, 13
- opus\_packet\_get\_samples\_per\_frame
  - Opus Decoder, 13
- opus\_packet\_parse
  - Opus Decoder, 14
- opus\_repacketizer\_cat
  - Repacketizer, 15
- opus\_repacketizer\_create
  - Repacketizer, 15
- opus\_repacketizer\_destroy
  - Repacketizer, 15
- opus\_repacketizer\_get\_nb\_frames
  - Repacketizer, 15
- opus\_repacketizer\_get\_size
  - Repacketizer, 15
- opus\_repacketizer\_init
  - Repacketizer, 15
- opus\_repacketizer\_out
  - Repacketizer, 15
- opus\_repacketizer\_out\_range
  - Repacketizer, 15
- OPUS\_RESET\_STATE
  - Generic CTLs, 27
- OPUS\_SET\_APPLICATION
  - Encoder related CTLs, 21
- OPUS\_SET\_BANDWIDTH
  - Encoder related CTLs, 22
- OPUS\_SET\_BITRATE
  - Encoder related CTLs, 22
- OPUS\_SET\_COMPLEXITY
  - Encoder related CTLs, 22
- OPUS\_SET\_DTX
  - Encoder related CTLs, 23
- OPUS\_SET\_FORCE\_CHANNELS
  - Encoder related CTLs, 23
- OPUS\_SET\_INBAND\_FEC
  - Encoder related CTLs, 23
- OPUS\_SET\_PACKET\_LOSS\_PERC
  - Encoder related CTLs, 23
- OPUS\_SET\_RESTRICTED\_LOWDELAY
  - Encoder related CTLs, 24
- OPUS\_SET\_SIGNAL
  - Encoder related CTLs, 24
- OPUS\_SET\_VBR
  - Encoder related CTLs, 24
- OPUS\_SET\_VBR\_CONSTRAINT

- Encoder related CTLs, [25](#)
- OPUS\_SET\_VOICE\_RATIO
  - Encoder related CTLs, [25](#)
- opus\_strerror
  - Opus library information functions, [27](#)
- opus\_types.h, [38](#)
  - opus\_int, [39](#)
  - opus\_int16, [39](#)
  - opus\_int32, [39](#)
  - opus\_int64, [39](#)
  - opus\_int8, [39](#)
  - opus\_uint, [39](#)
  - opus\_uint16, [39](#)
  - opus\_uint32, [39](#)
  - opus\_uint64, [39](#)
  - opus\_uint8, [39](#)
- opus\_uint
  - opus\_types.h, [39](#)
- opus\_uint16
  - opus\_types.h, [39](#)
- opus\_uint32
  - opus\_types.h, [39](#)
- opus\_uint64
  - opus\_types.h, [39](#)
- opus\_uint8
  - opus\_types.h, [39](#)
- OPUS\_UNIMPLEMENTED
  - Error codes, [16](#)
- OpusDecoder
  - Opus Decoder, [9](#)
- OpusEncoder
  - Opus Encoder, [6](#)
- OpusMSDecoder
  - opus\_multistream.h, [35](#)
- OpusMSEncoder
  - opus\_multistream.h, [35](#)
- OpusRepacketizer
  - Repacketizer, [15](#)
- Repacketizer, [14](#)
  - opus\_repacketizer\_cat, [15](#)
  - opus\_repacketizer\_create, [15](#)
  - opus\_repacketizer\_destroy, [15](#)
  - opus\_repacketizer\_get\_nb\_frames, [15](#)
  - opus\_repacketizer\_get\_size, [15](#)
  - opus\_repacketizer\_init, [15](#)
  - opus\_repacketizer\_out, [15](#)
  - opus\_repacketizer\_out\_range, [15](#)
  - OpusRepacketizer, [15](#)