

Opus

0.9.6-129-g0ca076d

Generated by Doxygen 1.7.4

Sat Sep 10 2011 20:04:40

Contents

1	Module Index	1
1.1	Modules	1
2	File Index	3
2.1	File List	3
3	Module Documentation	5
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	opus_decoder_destroy	11
3.2.2.6	opus_decoder_get_nb_samples	11
3.2.2.7	opus_decoder_get_size	12
3.2.2.8	opus_decoder_init	12
3.2.2.9	opus_packet_get_bandwidth	12
3.2.2.10	opus_packet_get_nb_channels	13
3.2.2.11	opus_packet_get_nb_frames	13
3.2.2.12	opus_packet_get_samples_per_frame	14
3.2.2.13	opus_packet_parse	14
3.3	Repacketizer	14
3.3.1	Typedef Documentation	15
3.3.1.1	OpusRepacketizer	15
3.3.2	Function Documentation	15
3.3.2.1	opus_repacketizer_cat	15
3.3.2.2	opus_repacketizer_create	15
3.3.2.3	opus_repacketizer_destroy	15
3.3.2.4	opus_repacketizer_get_nb_frames	15
3.3.2.5	opus_repacketizer_get_size	15
3.3.2.6	opus_repacketizer_init	15
3.3.2.7	opus_repacketizer_out	15
3.3.2.8	opus_repacketizer_out_range	15
3.4	Error codes	15
3.4.1	Define Documentation	16
3.4.1.1	OPUS_ALLOC_FAIL	16
3.4.1.2	OPUS_BAD_ARG	16
3.4.1.3	OPUS_BUFFER_TOO_SMALL	16
3.4.1.4	OPUS_INTERNAL_ERROR	16
3.4.1.5	OPUS_INVALID_PACKET	16
3.4.1.6	OPUS_INVALID_STATE	16
3.4.1.7	OPUS_OK	17
3.4.1.8	OPUS_UNIMPLEMENTED	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_BANDWIDTH	19
3.5.2.3	OPUS_GET_BITRATE	19
3.5.2.4	OPUS_GET_COMPLEXITY	19
3.5.2.5	OPUS_GET_DTX	19
3.5.2.6	OPUS_GET_FORCE_CHANNELS	20
3.5.2.7	OPUS_GET_INBAND_FEC	20
3.5.2.8	OPUS_GET_LOOKAHEAD	20
3.5.2.9	OPUS_GET_PACKET_LOSS_PERC	20
3.5.2.10	OPUS_GET_RESTRICTED_LOWDELAY	21
3.5.2.11	OPUS_GET_SIGNAL	21
3.5.2.12	OPUS_GET_VBR	21
3.5.2.13	OPUS_GET_VBR_CONSTRAINT	21
3.5.2.14	OPUS_GET_VOICE_RATIO	22
3.5.2.15	OPUS_SET_APPLICATION	22
3.5.2.16	OPUS_SET_BANDWIDTH	22
3.5.2.17	OPUS_SET_BITRATE	23
3.5.2.18	OPUS_SET_COMPLEXITY	23
3.5.2.19	OPUS_SET_DTX	23
3.5.2.20	OPUS_SET_FORCE_CHANNELS	23
3.5.2.21	OPUS_SET_INBAND_FEC	24
3.5.2.22	OPUS_SET_PACKET_LOSS_PERC	24
3.5.2.23	OPUS_SET_RESTRICTED_LOWDELAY	24
3.5.2.24	OPUS_SET_SIGNAL	24
3.5.2.25	OPUS_SET_VBR	25
3.5.2.26	OPUS_SET_VBR_CONSTRAINT	25
3.5.2.27	OPUS_SET_VOICE_RATIO	25
3.6	Generic CTLs	26
3.6.1	Detailed Description	26
3.6.2	Define Documentation	26
3.6.2.1	OPUS_GET_FINAL_RANGE	26
3.6.2.2	OPUS_RESET_STATE	26
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
4	File Documentation	29
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	Typedef Documentation	34
4.3.2.1	OpusMSDecoder	34
4.3.2.2	OpusMSEncoder	34
4.3.3	Function Documentation	34
4.3.3.1	opus_multistream_decode	34
4.3.3.2	opus_multistream_decode_float	35
4.3.3.3	opus_multistream_decoder_create	35
4.3.3.4	opus_multistream_decoder_ctl	35
4.3.3.5	opus_multistream_decoder_destroy	35
4.3.3.6	opus_multistream_decoder_init	35
4.3.3.7	opus_multistream_encode	36
4.3.3.8	opus_multistream_encode_float	36
4.3.3.9	opus_multistream_encoder_create	36
4.3.3.10	opus_multistream_encoder_ctl	36
4.3.3.11	opus_multistream_encoder_destroy	36
4.3.3.12	opus_multistream_encoder_init	37
4.4	opus_types.h File Reference	37
4.4.1	Detailed Description	37
4.4.2	Define Documentation	37
4.4.2.1	opus_int	37
4.4.2.2	opus_int64	37
4.4.2.3	opus_int8	37
4.4.2.4	opus_uint	38
4.4.2.5	opus_uint64	38

4.4.2.6	opus_uint8	38
4.4.3	Typedef Documentation	38
4.4.3.1	opus_int16	38
4.4.3.2	opus_int32	38
4.4.3.3	opus_uint16	38
4.4.3.4	opus_uint32	38

Chapter 1

Module Index

1.1 Modules

Here is a list of all modules:

Opus Encoder	5
Opus Decoder	8
Repacketizer	14
Error codes	15
Encoder related CTLs	17
Generic CTLs	26
Opus library information functions	27

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

opus.h (Opus reference implementation API)	29
opus_defines.h (Opus reference implementation constants)	31
opus_multistream.h (Opus reference implementation multistream API)	33
opus_types.h (Opus reference implementation types)	37

Chapter 3

Module Documentation

3.1 Opus Encoder

Typedefs

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

Functions

- OPUS_EXPORT int [opus_encoder_get_size](#) (int channels)
- OPUS_EXPORT [OpusEncoder](#) * [opus_encoder_create](#) ([opus_int32](#) Fs, int channels, int application, int *error)
Allocates and initializes an encoder state.
- OPUS_EXPORT 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](#).
- OPUS_EXPORT int [opus_encode](#) ([OpusEncoder](#) *st, const [opus_int16](#) *pcm, int frame_size, unsigned char *data, int max_data_bytes)
Encodes an Opus frame.
- OPUS_EXPORT 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.
- OPUS_EXPORT void [opus_encoder_destroy](#) ([OpusEncoder](#) *st)
Frees an OpusEncoder allocated by [opus_encoder_create](#).
- OPUS_EXPORT 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 OPUS_EXPORT 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 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 bi-rate

Returns

length of the data payload (in bytes)

3.1.2.2 OPUS_EXPORT 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 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 OPUS_EXPORT 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 OPUS_EXPORT 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 OPUS_EXPORT void opus_encoder_destroy (OpusEncoder * *st*)

Frees an OpusEncoder allocated by opus_encoder_create.

Parameters

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

3.1.2.6 OPUS_EXPORT int opus_encoder_get_size (int *channels*)

3.1.2.7 OPUS_EXPORT 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	<i>st</i>	OpusEncoder*: Encoder 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
in	<i>application</i>	int: Coding mode (OPUS_APPLICATION_VOIP/OPUS_APPLICATION_AUDIO/OPUS_APPLICATION_RESTRICTED_LOWDELAY)

Return values

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

3.2 Opus Decoder

Typedefs

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

Functions

- OPUS_EXPORT int [opus_decoder_get_size](#) (int *channels*)
Gets the size of an OpusDecoder structure.

- OPUS_EXPORT [OpusDecoder](#) * [opus_decoder_create](#) ([opus_int32](#) Fs, int channels, int *error)
Allocates and initializes a decoder state.
- OPUS_EXPORT int [opus_decoder_init](#) ([OpusDecoder](#) *st, [opus_int32](#) Fs, int channels)
Initializes a previously allocated decoder state.
- OPUS_EXPORT int [opus_decode](#) ([OpusDecoder](#) *st, const unsigned char *data, int len, [opus_int16](#) *pcm, int frame_size, int decode_fec)
Decode an Opus frame.
- OPUS_EXPORT 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.
- OPUS_EXPORT int [opus_decoder_ctl](#) ([OpusDecoder](#) *st, int request,...)
Perform a CTL function on an Opus decoder.
- OPUS_EXPORT void [opus_decoder_destroy](#) ([OpusDecoder](#) *st)
Frees an OpusDecoder allocated by opus_decoder_create.
- OPUS_EXPORT 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_EXPORT int [opus_packet_get_bandwidth](#) (const unsigned char *data)
Gets the bandwidth of an Opus packet.
- OPUS_EXPORT 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.
- OPUS_EXPORT int [opus_packet_get_nb_channels](#) (const unsigned char *data)
Gets the number of channels from an Opus packet.
- OPUS_EXPORT int [opus_packet_get_nb_frames](#) (const unsigned char packet[], int len)
Gets the number of frame in an Opus packet.
- OPUS_EXPORT 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 `OPUS_EXPORT 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 `OPUS_EXPORT 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 OPUS_EXPORT 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 OPUS_EXPORT int opus_decoder_ctl (OpusDecoder* *st*, int *request*, ...)

Perform a CTL function on an Opus decoder.

See also

decoderctls

3.2.2.5 OPUS_EXPORT 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 OPUS_EXPORT 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 OPUS_EXPORT 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 OPUS_EXPORT 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 OPUS_EXPORT 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 OPUS_EXPORT 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 OPUS_EXPORT 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 OPUS_EXPORT 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

OPUS_INVALID_PACKET	The compressed data passed is corrupted or of an unsupported type
---------------------	---

3.2.2.13 OPUS_EXPORT 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

- OPUS_EXPORT int [opus_repacketizer_get_size](#) (void)
- OPUS_EXPORT [OpusRepacketizer](#) * [opus_repacketizer_init](#) ([OpusRepacketizer](#) *rp)
- OPUS_EXPORT [OpusRepacketizer](#) * [opus_repacketizer_create](#) (void)
- OPUS_EXPORT void [opus_repacketizer_destroy](#) ([OpusRepacketizer](#) *rp)
- OPUS_EXPORT int [opus_repacketizer_cat](#) ([OpusRepacketizer](#) *rp, const unsigned char *data, int len)
- OPUS_EXPORT int [opus_repacketizer_out_range](#) ([OpusRepacketizer](#) *rp, int begin, int end, unsigned char *data, int maxlen)
- OPUS_EXPORT int [opus_repacketizer_get_nb_frames](#) ([OpusRepacketizer](#) *rp)
- OPUS_EXPORT 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 OPUS_EXPORT int [opus_repacketizer_cat](#) ([OpusRepacketizer](#) * *rp*, const unsigned char * *data*, int *len*)

3.3.2.2 OPUS_EXPORT [OpusRepacketizer](#)* [opus_repacketizer_create](#) (void)

3.3.2.3 OPUS_EXPORT void [opus_repacketizer_destroy](#) ([OpusRepacketizer](#) * *rp*)

3.3.2.4 OPUS_EXPORT int [opus_repacketizer_get_nb_frames](#) ([OpusRepacketizer](#) * *rp*)

3.3.2.5 OPUS_EXPORT int [opus_repacketizer_get_size](#) (void)

3.3.2.6 OPUS_EXPORT [OpusRepacketizer](#)* [opus_repacketizer_init](#) ([OpusRepacketizer](#) * *rp*)

3.3.2.7 OPUS_EXPORT int [opus_repacketizer_out](#) ([OpusRepacketizer](#) * *rp*, unsigned char * *data*, int *maxlen*)

3.3.2.8 OPUS_EXPORT 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_GET_BANDWIDTH\(x\)](#)
Gets the encoder's configured 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

[opus_encoder_ctl](#)

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_BANDWIDTH(x)

Gets the encoder's configured bandpass,.

See also

[OPUS_SET_BANDWIDTH](#)

Parameters

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

3.5.2.3 #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.
-----	---	--

3.5.2.4 #define OPUS_GET_COMPLEXITY(x)

Gets the encoder's complexity configuration,.

See also

[OPUS_SET_COMPLEXITY](#)

Parameters

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

3.5.2.5 #define OPUS_GET_DTX(x)

Gets encoder's configured use of discontinuous transmission,.

See also

[OPUS_SET_DTX](#)

Parameters

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

3.5.2.6 #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
-----	---	-----------------------

3.5.2.7 #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
-----	---	----------------

3.5.2.8 #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
-----	---	-----------------------------------

3.5.2.9 #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.
-----	---	--

3.5.2.10 #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
-----	---	------------

3.5.2.11 #define OPUS_GET_SIGNAL(x)

Gets the encoder's configured signal type,.

See also

[OPUS_SET_SIGNAL](#)

Parameters

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

3.5.2.12 #define OPUS_GET_VBR(x)

Gets the encoder's VBR configuration,.

See also

[OPUS_SET_VBR](#)

Parameters

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

3.5.2.13 #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
-----	---	------------

3.5.2.14 #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.
-----	---	---

3.5.2.15 #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
----	---	------------------------

3.5.2.16 #define OPUS_SET_BANDWIDTH(x)

Configures the encoder's bandpass.

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
----	---	----------------------

3.5.2.17 #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.
----	---	---

3.5.2.18 #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 inconsistent between modes

Parameters

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

3.5.2.19 #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
----	---	--

3.5.2.20 #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)
----	---	--

3.5.2.21 `#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
----	---	--

3.5.2.22 `#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.
----	---	---

3.5.2.23 `#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)
----	---	--------------------------------

3.5.2.24 `#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
----	---	------------------

3.5.2.25 #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)
----	---	---------------------

3.5.2.26 #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
----	---	---------------------

3.5.2.27 #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.

3.6.1 Detailed Description

See also

[opus_encoder_ctl](#), [opus_decoder_ctl](#)

3.6.2 Define Documentation

3.6.2.1 `#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_int32*: Entropy coder state
-----	---	----------------------------------

3.6.2.2 `#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

- OPUS_EXPORT const char * [opus_strerror](#) (int error)
Converts an opus error code into a human readable string.
- OPUS_EXPORT const char * [opus_get_version_string](#) (void)
Gets the libopus version string.

3.7.1 Function Documentation

3.7.1.1 OPUS_EXPORT const char* opus_get_version_string (void)

Gets the libopus version string.

Returns

Version string

3.7.1.2 OPUS_EXPORT const char* opus_strerror (int error)

Converts an opus error code into a human readable string.

Parameters

in	<i>error</i>	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

- OPUS_EXPORT int [opus_encoder_get_size](#) (int channels)
- OPUS_EXPORT [OpusEncoder](#) * [opus_encoder_create](#) ([opus_int32](#) Fs, int channels, int application, int *error)
Allocates and initializes an encoder state.
- OPUS_EXPORT 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.
- OPUS_EXPORT int [opus_encode](#) ([OpusEncoder](#) *st, const [opus_int16](#) *pcm, int frame_size, unsigned char *data, int max_data_bytes)
Encodes an Opus frame.
- OPUS_EXPORT 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.

- OPUS_EXPORT void [opus_encoder_destroy](#) ([OpusEncoder](#) *st)

Frees an OpusEncoder allocated by opus_encoder_create.

- OPUS_EXPORT int [opus_encoder_ctl](#) ([OpusEncoder](#) *st, int request,...)

Perform a CTL function on an Opus encoder.

- OPUS_EXPORT int [opus_decoder_get_size](#) (int channels)

Gets the size of an OpusDecoder structure.

- OPUS_EXPORT [OpusDecoder](#) * [opus_decoder_create](#) ([opus_int32](#) Fs, int channels, int *error)

Allocates and initializes a decoder state.

- OPUS_EXPORT int [opus_decoder_init](#) ([OpusDecoder](#) *st, [opus_int32](#) Fs, int channels)

Initializes a previously allocated decoder state.

- OPUS_EXPORT int [opus_decode](#) ([OpusDecoder](#) *st, const unsigned char *data, int len, [opus_int16](#) *pcm, int frame_size, int decode_fec)

Decode an Opus frame.

- OPUS_EXPORT 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.

- OPUS_EXPORT int [opus_decoder_ctl](#) ([OpusDecoder](#) *st, int request,...)

Perform a CTL function on an Opus decoder.

- OPUS_EXPORT void [opus_decoder_destroy](#) ([OpusDecoder](#) *st)

Frees an OpusDecoder allocated by opus_decoder_create.

- OPUS_EXPORT 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_EXPORT int [opus_packet_get_bandwidth](#) (const unsigned char *data)

Gets the bandwidth of an Opus packet.

- OPUS_EXPORT 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.

- OPUS_EXPORT int [opus_packet_get_nb_channels](#) (const unsigned char *data)

Gets the number of channels from an Opus packet.

- OPUS_EXPORT int [opus_packet_get_nb_frames](#) (const unsigned char packet[], int len)

Gets the number of frame in an Opus packet.

- OPUS_EXPORT int [opus_decoder_get_nb_samples](#) (const [OpusDecoder](#) *dec, const unsigned char packet[], int len)

Gets the number of samples of an Opus packet.

- OPUS_EXPORT int [opus_repacketizer_get_size](#) (void)
- OPUS_EXPORT [OpusRepacketizer](#) * [opus_repacketizer_init](#) ([OpusRepacketizer](#) *rp)
- OPUS_EXPORT [OpusRepacketizer](#) * [opus_repacketizer_create](#) (void)

- OPUS_EXPORT void [opus_repacketizer_destroy](#) (OpusRepacketizer *rp)
- OPUS_EXPORT int [opus_repacketizer_cat](#) (OpusRepacketizer *rp, const unsigned char *data, int len)
- OPUS_EXPORT int [opus_repacketizer_out_range](#) (OpusRepacketizer *rp, int begin, int end, unsigned char *data, int maxlen)
- OPUS_EXPORT int [opus_repacketizer_get_nb_frames](#) (OpusRepacketizer *rp)
- OPUS_EXPORT 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_GET_BANDWIDTH\(x\)](#)
Gets the encoder's configured 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.

Functions

- OPUS_EXPORT const char * [opus_strerror](#) (int error)

Converts an opus error code into a human readable string.
- OPUS_EXPORT 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"
```

Typedefs

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

Functions

- OPUS_EXPORT [OpusMSEncoder](#) * [opus_multistream_encoder_create](#) ([opus_int32](#) Fs, int channels, int streams, int coupled_streams, unsigned char *mapping, int application, int *error)
- OPUS_EXPORT int [opus_multistream_encoder_init](#) ([OpusMSEncoder](#) *st, [opus_int32](#) Fs, int channels, int streams, int coupled_streams, unsigned char *mapping, int application)

- OPUS_EXPORT 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)
- OPUS_EXPORT 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)
- OPUS_EXPORT void [opus_multistream_encoder_destroy](#) ([OpusMSEncoder](#) *st)
- OPUS_EXPORT int [opus_multistream_encoder_ctl](#) ([OpusMSEncoder](#) *st, int request,...)
- OPUS_EXPORT [OpusMSDecoder](#) * [opus_multistream_decoder_create](#) ([opus_int32](#) Fs, int channels, int streams, int coupled_streams, unsigned char *mapping, int *error)
- OPUS_EXPORT int [opus_multistream_decoder_init](#) ([OpusMSDecoder](#) *st, [opus_int32](#) Fs, int channels, int streams, int coupled_streams, unsigned char *mapping)
- OPUS_EXPORT 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.
- OPUS_EXPORT 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.
- OPUS_EXPORT int [opus_multistream_decoder_ctl](#) ([OpusMSDecoder](#) *st, int request,...)
- OPUS_EXPORT void [opus_multistream_decoder_destroy](#) ([OpusMSDecoder](#) *st)

4.3.1 Detailed Description

Opus reference implementation multistream API.

4.3.2 Typedef Documentation

4.3.2.1 typedef struct [OpusMSDecoder](#) [OpusMSDecoder](#)

4.3.2.2 typedef struct [OpusMSEncoder](#) [OpusMSEncoder](#)

4.3.3 Function Documentation

4.3.3.1 OPUS_EXPORT 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 (interleaved if 2 channels). 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.

4.3.3.2 **OPUS_EXPORT** 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. Use a NULL pointer to indicate packet loss
<i>len</i>	Number of bytes in payload
<i>pcm</i>	Output signal (interleaved if 2 channels). 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.

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

Parameters

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

4.3.3.4 **OPUS_EXPORT** int opus_multistream_decoder_ctl (**OpusMSDecoder** * *st*, int *request*, ...)

4.3.3.5 **OPUS_EXPORT** void opus_multistream_decoder_destroy (**OpusMSDecoder** * *st*)

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

Parameters

<i>st</i>	Encoder state
<i>Fs</i>	Sampling rate of input signal (Hz)
<i>channels</i>	Number of channels (1/2) in input signal

4.3.3.7 **OPUS_EXPORT** 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)

Parameters

<i>st</i>	Encoder state
<i>pcm</i>	Input signal (interleaved if 2 channels). length is frame_size*channels
<i>frame_size</i>	Number of samples per frame of input signal
<i>data</i>	Output payload (no more than max_data_bytes long)
<i>max_data_bytes</i>	Allocated memory for payload; don't use for controlling bitrate

4.3.3.8 **OPUS_EXPORT** 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)

Parameters

<i>st</i>	Encoder state
<i>pcm</i>	Input signal (interleaved if 2 channels). length is frame_size*channels
<i>frame_size</i>	Number of samples per frame of input signal
<i>data</i>	Output payload (no more than max_data_bytes long)
<i>max_data_bytes</i>	Allocated memory for payload; don't use for controlling bitrate

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

Parameters

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

4.3.3.10 **OPUS_EXPORT** int opus_multistream_encoder_ctl (**OpusMSEncoder** * *st*, int *request*, ...)

4.3.3.11 **OPUS_EXPORT** void opus_multistream_encoder_destroy (**OpusMSEncoder** * *st*)

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

Parameters

<i>st</i>	Encoder state
<i>Fs</i>	Sampling rate of input signal (Hz)
<i>channels</i>	Number of channels (1/2) in input signal
<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

Encoder related CTLs, [17](#)

- OPUS_GET_APPLICATION, [18](#)
- OPUS_GET_BANDWIDTH, [18](#)
- OPUS_GET_BITRATE, [19](#)
- OPUS_GET_COMPLEXITY, [19](#)
- OPUS_GET_DTX, [19](#)
- OPUS_GET_FORCE_CHANNELS, [19](#)
- OPUS_GET_INBAND_FEC, [20](#)
- OPUS_GET_LOOKAHEAD, [20](#)
- OPUS_GET_PACKET_LOSS_PERC, [20](#)
- OPUS_GET_RESTRICTED_LOWDELAY, [21](#)
- OPUS_GET_SIGNAL, [21](#)
- OPUS_GET_VBR, [21](#)
- OPUS_GET_VBR_CONSTRAINT, [21](#)
- OPUS_GET_VOICE_RATIO, [22](#)
- OPUS_SET_APPLICATION, [22](#)
- OPUS_SET_BANDWIDTH, [22](#)
- OPUS_SET_BITRATE, [23](#)
- OPUS_SET_COMPLEXITY, [23](#)
- OPUS_SET_DTX, [23](#)
- OPUS_SET_FORCE_CHANNELS, [23](#)
- OPUS_SET_INBAND_FEC, [23](#)
- OPUS_SET_PACKET_LOSS_PERC, [24](#)
- OPUS_SET_RESTRICTED_LOWDELAY, [24](#)
- OPUS_SET_SIGNAL, [24](#)
- OPUS_SET_VBR, [25](#)
- 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, [17](#)

Generic CTLs, [26](#)

- OPUS_GET_FINAL_RANGE, [26](#)
- OPUS_RESET_STATE, [26](#)

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
 - Encoder related CTLs, [18](#)
- OPUS_GET_BITRATE
 - Encoder related CTLs, [19](#)
- 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, [20](#)
- OPUS_GET_LOOKAHEAD
 - Encoder related CTLs, [20](#)
- OPUS_GET_PACKET_LOSS_PERC
 - Encoder related CTLs, [20](#)
- OPUS_GET_RESTRICTED_LOWDELAY
 - Encoder related CTLs, [21](#)
- OPUS_GET_SIGNAL
 - Encoder related CTLs, [21](#)
- 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, [22](#)
- opus_int
 - opus_types.h, [37](#)
- opus_int16
 - opus_types.h, [38](#)
- opus_int32
 - opus_types.h, [38](#)
- opus_int64
 - opus_types.h, [37](#)
- opus_int8
 - opus_types.h, [37](#)
- OPUS_INTERNAL_ERROR
 - Error codes, [16](#)
- OPUS_INVALID_PACKET
 - Error codes, [16](#)
- OPUS_INVALID_STATE
 - Error codes, [16](#)
- opus_multistream.h, [33](#)
 - opus_multistream_decode, [34](#)
 - opus_multistream_decode_float, [35](#)
 - opus_multistream_decoder_create, [35](#)
 - opus_multistream_decoder_ctl, [35](#)
 - opus_multistream_decoder_destroy, [35](#)
 - opus_multistream_decoder_init, [35](#)
 - opus_multistream_encode, [35](#)
 - opus_multistream_encode_float, [36](#)
 - opus_multistream_encoder_create, [36](#)
 - opus_multistream_encoder_ctl, [36](#)
 - opus_multistream_encoder_destroy, [36](#)
 - opus_multistream_encoder_init, [36](#)
 - OpusMSDecoder, [34](#)
 - OpusMSEncoder, [34](#)
- opus_multistream_decode
 - opus_multistream.h, [34](#)
- opus_multistream_decode_float
 - opus_multistream.h, [35](#)
- opus_multistream_decoder_create
 - opus_multistream.h, [35](#)
- opus_multistream_decoder_ctl
 - opus_multistream.h, [35](#)

- opus_multistream_decoder_destroy
 - opus_multistream.h, [35](#)
- opus_multistream_decoder_init
 - opus_multistream.h, [35](#)
- opus_multistream_encode
 - opus_multistream.h, [35](#)
- opus_multistream_encode_float
 - opus_multistream.h, [36](#)
- opus_multistream_encoder_create
 - opus_multistream.h, [36](#)
- opus_multistream_encoder_ctl
 - opus_multistream.h, [36](#)
- opus_multistream_encoder_destroy
 - opus_multistream.h, [36](#)
- opus_multistream_encoder_init
 - opus_multistream.h, [36](#)
- 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, [26](#)
- OPUS_SET_APPLICATION
 - Encoder related CTLs, [22](#)
- OPUS_SET_BANDWIDTH
 - Encoder related CTLs, [22](#)
- OPUS_SET_BITRATE
 - Encoder related CTLs, [23](#)
- OPUS_SET_COMPLEXITY
 - Encoder related CTLs, [23](#)
- 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, [24](#)
- OPUS_SET_RESTRICTED_LOWDELAY
 - Encoder related CTLs, [24](#)
- OPUS_SET_SIGNAL
 - Encoder related CTLs, [24](#)
- OPUS_SET_VBR
 - Encoder related CTLs, [25](#)
- 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, [37](#)
 - opus_int, [37](#)
 - opus_int16, [38](#)
 - opus_int32, [38](#)
 - opus_int64, [37](#)
 - opus_int8, [37](#)
 - opus_uint, [37](#)
 - opus_uint16, [38](#)
 - opus_uint32, [38](#)
 - opus_uint64, [38](#)
 - opus_uint8, [38](#)
- opus_uint
 - opus_types.h, [37](#)
- opus_uint16
 - opus_types.h, [38](#)
- opus_uint32
 - opus_types.h, [38](#)
- opus_uint64
 - opus_types.h, [38](#)
- opus_uint8
 - opus_types.h, [38](#)
- OPUS_UNIMPLEMENTED
 - Error codes, [17](#)
- OpusDecoder
 - Opus Decoder, [9](#)
- OpusEncoder

- Opus Encoder, [6](#)
- OpusMSDecoder
 - opus_multistream.h, [34](#)
- OpusMSEncoder
 - opus_multistream.h, [34](#)
- 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](#)