OPUS Codec Application Note

Version: 1.0
Release date: 17 January 2019

Document revision history

Revision	Date	Description
1.0	17 January 2019	Initial release

Table of contents

Lists of tables and figure

1. Introduction

An ogg API provides the encoder and decoder. The ogg uses a Constrained Energy Lapped Transform (CELT) encoder and CELT decoder to implement the codec. The CELT bitstream must be packed as an ogg file.

The ogg codec API includes the following functions:

(1) int OGGCELT_init (char *drate, char *c, char *out, ogg_enc_mem_t *ogg_enc, OpusCustomEncoder **enc). The function is for the initialization of an ogg encoder.

Table 1. The information of OGG encoder initialized

Functionality	Description
Parameters	[in] **drate** is the compressed bitrate of an encoder that provides support for a range between 16 and 64 Kbps. The drate variable is a string type variable. For example, drate = “16” selects 16 Kbps option. [in] **c** is the complexity of the encoder that support range from 0 (low quality) to 3 (high quality). The c variable is character type. For example, c = “3” selects high quality option. [out] **out** is buffer pointer needs about 4K bytes space to store ogg compressed bitstream data. [in] **ogg_enc** is an ogg_enc_mem_t structure pointer that creates the memory of an ogg bitstream packer. [in] **enc** is double pointer (i.e. pointer to pointer) of the OpusCustomEncoder structure that creates the memory of CELT encoder.
Returns	-1: the operation failed >0: the size of the ogg header

(2) int OGGCELT_Proc(short *pcm, unsigned char *compressed, int nbsamples, ogg_enc_mem_t *ogg_enc, OpusCustomEncoder *enc); The function should be called when the ogg encoder is processing.

Table 2. The information of OGG encoder process

Functionality	Description
Parameters	[in] **pcm** is is input buffer pointer for the CELT encoder. [out] **compressed** Is buffer pointer needs about 4K bytes buffer space to store the ogg CELT compressed bitstream data. [in] **nbsamples** is the length of pcm input buffer. The pcm length is 640 bytes @ 16K sample rate. [in] **ogg_enc** is a ogg_enc_mem_t structure pointer that processes the memory of an ogg bitstream packer. [in] **enc** is a OpusCustomEncoder structure pointer that processes the memory of the CELT encoder.
Returns	-1: the end of the ogg CELT encoder !=0: the process status of the ogg CELT encoder

(3) void OGGCELT_uninit (ogg_enc_mem_t *ogg_mem, OpusCustomEncoder *enc); The ogg function should be called when the opus encoder finishes freeing the ogg encoder memory.

Table 3. The information of OGG encoder uninitialized

Functionality	Description
Parameters	[in] **ogg_mem** is a ogg_enc_mem_t structure pointer that frees the memory of the ogg bitstream packer. [in] **enc** is a OpusCustomEncoder structure pointer that frees the memory of a CELT encoder.
Returns	=0: the operation successfully completed !=0: the operation failed

(4) int OGG_PARSER_INIT(ogg_dec_mem_t *ogg_mem); The function is the initialization of the ogg parser. This parser can extract the CELT bitstream from an ogg file (or bitstream).

Table 4. The information of OGG parser initialized

Functionality	Description
Parameters	[in] **ogg_mem** is a ogg_dec_mem_t structure pointer that creates the memory of an ogg bitstream parser.
Returns	=0: the operation successfully completed !=0: the operation failed

(5) int OGG_PARSER_PROC(char *buf_in, ogg_dec_mem_t *ogg_mem); The function should be called when processing the parser of the ogg. The ogg bitstream data uses this API to unpack the CELT bitstream and then feed the results to the OPUSCELT_DEC_16K_C1_F320_proc API to get the PCM output

Table 5. The information of OGG parser unpack bit stream

Functionality	Description
Parameters	[in] **buf_in** is input buffer pointer for ogg bit stream data. [in] **ogg_mem** is a ogg_dec_mem_t structure pointer that unpack ogg bit stream.
Returns	=0: the operation successfully completed !=0: the operation failed

(6) void OGG_PARSER_UNINIT(ogg_dec_mem_t *ogg_mem); The ogg function should be called when the ogg parser finishes freeing the memory of ogg_mem.

Table 6. The information of OGG parser uninitialized

Functionality	Description
Parameters	[in] **ogg_mem** is a ogg_dec_mem_t structure pointer that frees the memory of the ogg bitstream parser.
Returns	None

(7) int OPUSCELT_DEC_16K_C1_F320_init(OpusCustomDecoder **avc_dec); The function is the initialization of the CELT decoder.

Table 7. The information of the CELT decoder initialized

Functionality	Description
Parameters	[in] **avc_dec** is a OpusCustomDecoder structure pointer that creates the memory of the CELT decoder.
Returns	=0: the operation successfully completed !=0: the operation failed

(8) int OPUSCELT_DEC_16K_C1_F320_proc(unsigned char *compressed, short *pcm, int nbCompressedBytes, OpusCustomDecoder *avc_dec); The function should be called when the CELT decoder is processed. The CELT bitstream comes from the OGG_PARSER_PROC API parser, and is then fed into this API to get the final PCM output.

Table 8. The information of CELT decoder process

Functionality	Description
Parameters	[in] **compressed** is the buffer pointer of CELT decoder bitstream. [out] **pcm** is the buffer pointer of the CELT decoder output. [in] **nbCompressedBytes** is the length of the CELT decoder bitstream. [in] **avc_dec** is a OpusCustomDecoder structure pointer that processes the memory of the CELT decoder.
Returns	The length of the PCM output by the ogg decoder.

(9) void OPUSCELT_DEC_uninit(OpusCustomDecoder *avc_dec); The ogg function should be called when the CELT decoder finishes freeing the CELT decoder memory.

Table 9. The information of CELT decoder uninitialized

Functionality	Description
Parameters	[in] **avc_dec** is a OpusCustomDecoder structure pointer that frees the CELT decoder memory.
Returns	None

2. Examples

How to use the OGG encoder:

Step 1: Set up the parameters, ogg packer, and the CELT encoder structure.

Sample code:

```
#define MAXSIZE          4000
#define Frame_size       640
ogg_enc_mem_t            ogg_enc;
OpusCustomEncoder        *ogg_celt_enc;
char                     input[Frame_size];
char                     buf[MAXSIZE];
```

Step 2: Call the OGGCELT_init API to set the sample rate to 32K and the complexity to 3.

Sample code:

```
ret = OGGCELT_init("32", "3", buf, &ogg_enc, &ogg_celt_enc);
```

Step 3: Read 640 bytes of PCM data from the PCM file.

Sample code:

```
fread(input, sizeof(char), Frame_size, fi);
```

Step 4: Call the OGGCELT_Proc API to run the OGG encoder.

Sample code:

```
ret = OGGCELT_Proc(input, buf, Frame_size, &ogg_enc, ogg_celt_enc);
// input        = 640 bytes PCM input buffer.
// buf          = ogg encoder packer bit stream output
// ogg_enc      = ogg_enc_mem_t structure
// ogg_celt_enc = OpusCustomEncoder structure
```

Step 5: If the OGG encoder still has PCM input data, go to **Step 4** in this procedure. Otherwise, go to **Step 6**.

Step 6: When the opus encoder is complete, the OGGCELT_Uninit API must be called.

Sample code:

```
OGGCELT_Uninit(&ogg_enc,ogg_celt_enc);
```

How to use the OGG decoder:

Step 1: Set up the parameters, ogg parser, and CELT decoder structure.

Sample code:

```
#define OPUS_BufInSize  256
OpusCustomDecoder       *ogg_celt_dec;
ogg_dec_mem_t           ogg_mem;
char                    buf_in[OPUS_BufInSize];
short                   buf_ou[OPUS_FrameeSize];
int                     status;
```

Step 2: Call the OPUSCELT_DEC_16K_C1_F320_init API to initialize the ogg decoder.

Sample code:

```
status = OPUSCELT_DEC_16K_C1_F320_init(&ogg_celt_dec);
```

Step 3: Call the OGG_PARSER_INIT API to initialize the ogg parser.

Sample code:

```
status = 0 OGG_PARSER_INIT(&ogg_mem);
```

Step 4: The buf_in buffer reads the 256-byte bitstream data from the ogg file (i.e. bitstream).

Step 5: Call the OGG_PARSER_PROC API to hander the parser initialization. If the status is 0 (as shown in the results of this API), the ogg parser is ready and you can go to Step 6 of this procedure. Otherwise, you must go back to Step 4 and get the next buf_in data.

Sample code:

```
status = OGG_PARSER_PROC(buf_in, &ogg_mem);
if (status == 0) goto Step6 ; else goto Step 4;
```

Step 6: Use OGG_PARSER_PROC to get the CELT bitstream and then fed this into the OPUSCELT_DEC_16K_C1_F320_proc API to create the final PCM output.

Sample code:

```
status = OGG_PARSER_PROC(buf_in, &ogg_mem);
    // buf_in            = input buffer from ogg file(bit stream).
    // ogg_celt_dec      = OpusCustomDecoder structure
status = OPUSCELT_DEC_16K_C1_F320_proc(ogg_mem.op.packet, buf_ou,
          ogg_mem.op.bytes, ogg_celt_dec);
    // ogg_mem.op.packet = celt decoder bit stream
    // buf_ou            = the pcm output of celt decoder
    // ogg_mem.op.bytes  = the byte length of celt bit stream
    // ogg_celt_dec      = OpusCustomDecoder structure
```

Step 7: If the ogg decoder still has CELT bitstream data, you must go back to Step 6 of this procedure. Otherwise, go to Step 8.

Step 8: When the ogg decoder is finished, you must call both the OPUSCELT_DEC_uninit and OGG_PARSER_UNINIT APIs.

Sample code:

```
OPUSCELT_DEC_uninit(ogg_celt_dec);
OGG_PARSER_UNINIT(&ogg_mem);
```