TS 1xx xxx V0.1.3 (2015-10)

Opus Interactive Audio Codec
Transport Multiplexing Standard

Reference
<Workitem>

Keywords
audio, broadcasting, coding, digital

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - NAF 742 C
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° 7803/88

Important notice

Individual copies of the present document can be downloaded from:
http://www.etsi.org

The present document may be made available in more than one electronic version or in print. In any case of existing or perceived difference in contents between such versions, the reference version is the Portable Document Format (PDF). In case of dispute, the reference shall be the printing on ETSI printers of the PDF version kept on a specific network drive within ETSI Secretariat.

Users of the present document should be aware that the document may be subject to revision or change of status.
Information on the current status of this and other ETSI documents is available at
http://portal.etsi.org/tb/status/status.asp

If you find errors in the present document, please send your comment to one of the following services:
http://portal.etsi.org/chaircor/ETSI_support.asp

Copyright Notification

Contents

<PAGE BREAK>

Intellectual Property Rights

IPRs essential or potentially essential to the present document may have been declared to ETSI. The information pertaining to these essential IPRs, if any, is publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (http://ipr.etsi.org).

Pursuant to the ETSI IPR Policy, no investigation, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Foreword

This Technical Specification (TS) has been produced by Joint Technical Committee (JTC) Broadcast of the European Broadcasting Union (EBU), Comité Européen de Normalisation ELECtrotechnique (CENELEC) and the European Telecommunications Standards Institute (ETSI).

Introduction

This document specifies how to combine one or more Opus elementary streams into a System A (Advanced Television Systems Committee (ATSC), ITU-R Recommendation BT.1300) or System B (Digital Video Brodcasting (DVB), ITU-R Recommendation BT.1300) Motion Picture Experts Group (MPEG) 2 transport stream (ISO/IEC 13818-1 [i.1]).

An Opus bitstream is multiplexed into an MPEG-2 transport stream like any other audio codec, by packetizing it into Packetized Elementary Stream (PES) packets. This document defines the codes necessary to unambiguously indentify an Opus stream and the audio descriptor needed to describe the contents of the bit stream in the Program-Specific Information (PSI) tables.

This includes stream_type, stream_id, an opus_audio_descriptor, and for System A, a registration_descriptor. opus_audio_descriptor serves as the public registration in System B. A standard ISO_639_language_descriptor may indicate language [i.1]. A single Opus frame can only encode one or two channels. These descriptors specify how to encode multichannel through the aggregation of multiple Opus streams into a single elementary stream. Some additional constraints are placed on the PES layer to allow decoding multiple audio streams in exact sample synchronization.

Check http://portal.etsi.org/edithelp/Files/other/EDRs_navigator.chm clauses 5.2.3 and A.4 for help.

<PAGE BREAK>

1 Scope

This document specifies how to multiplex Opus audio data [1] into an MPEG-2 transport stream. Opus audio data is suitable for digital audio transmission, storage, and interactive applications. Opus may convey up to 255 channels, coupled in pairs, with dynamic audio bandwidths from narrowband to full band and dynamic frame sizes that vary between 2.5 ms and 60 ms, at dynamic bitrates from 6 kbps to 255 kbps per channel, using both linear prediction (LP) for high-quality speech and the Modified Discrete Cosine Transform (MDCT) for high-quality music and other audio.

2 References

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

Referenced documents which are not found to be publicly available in the expected location might be found at http://docbox.etsi.org/Reference.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long term validity.

2.1 Normative references

The following referenced documents are necessary for the application of the present document.

[1] IETF RFC 6716: "Definition of the Opus Audio Codec".

[2] ETSI EN 300 163: "<Title>".

2.2 Informative references

The following referenced documents are not necessary for the application of the present document but they assist the user with regard to a particular subject area.

[i.1] ISO/IEC 13818-1: "Information technology – Generic coding of moving pictures and associated audio information: Systems".

[i.2] IETF draft-ietf-codec-oggopus: "Ogg Encapsulation for the Opus Audio Codec".

<PAGE BREAK>

3 Definitions, symbols and abbreviations

Delete from the above heading the word(s) which is/are not applicable, (see clauses 13 and 14 of EDRs).

*Definitions and abbreviations extracted from ETSI deliverables can be useful when drafting documents and can be consulted via the **Terms and Definitions Interactive Database (TEDDI)** (http://webapp.etsi.org/Teddi/).*

3.1 Definitions

Clause numbering depends on applicability.

- ***A definition shall not take the form of, or contain, a requirement.***
- ***The form of a definition shall be such that it can replace the term in context. Additional information shall be given only in the form of examples or notes (see below).***
- ***The terms and definitions shall be presented in alphabetical order.***

For the purposes of the present document, the [following] terms and definitions [given in ... and the following] apply:

Definition format

<defined term>: <definition>

example 1: text used to clarify abstract rules by applying them literally

NOTE: This may contain additional information.

3.2 Symbols

Clause numbering depends on applicability.

For the purposes of the present document, the [following] symbols [given in ... and the following] apply:

Symbol format

<symbol> <Explanation>
<2nd symbol> <2nd Explanation>
<3rd symbol> <3rd Explanation>

3.3 Abbreviations

Abbreviations should be ordered alphabetically.

Clause numbering depends on applicability.

For the purposes of the present document, the [following] abbreviations [given in ... and the following] apply:

Abbreviation format

<ACRONYM1> <Explanation>
<ACRONYM2> <Explanation>
<ACRONYM3> <Explanation>

4 Detailed Specification for System A (ATSC)

4.1 stream_type

The value of stream_type for Opus shall be 0×??. [TODO: 0×88 appears next on the list. Can we share with DVB?]

4.2 stream_id

The value of stream_id in the PES header shall be 0×BD (indicating private_stream_1). Multiple Opus streams may share the same value of stream_id since each stream is carried with a unique packet identifier (PID) value. The mapping of values of PID to stream_type is indicated in the transport stream Program Map Table (PMT).

4.3 registration_descriptor

The syntax of the ISO/IEC 13818-1 [i.1] registration_descriptor for Opus streams is shown in Table 4 -1.

Table 4-1 Opus registration_descriptor syntax

Syntax	Number of bits	Identifier
registration_descriptor() {		
descriptor_tag	**8**	**uimsbf**
descriptor_length	**8**	**uimsbf**
format_identifier	**32**	**uimsbf**
}		

4.3.1 Semantics for the Opus registration_descriptor

descriptor_tag: The descriptor tag is an 8-bit field which identifies each descriptor. The value of the tag for the registration_descriptor is 0×05.

descriptor_length: This 8-bit field specifies the total number of bytes of the data portion of the descriptor following the byte defining the value of this field. The value of this field for the Opus registration_descriptor is 0×04.

format_identifier: The format_identifier is a 32-bit value obtained from a Registration Authority as designated by ISO/IEC JTC 1/SC 29. The value of this field for the Opus registration_descriptor is 0×4F707573 (“Opus”). [TODO: Actually register this: http://smpte-ra.org/mpegreg/mpeg.html]

4.4 opus_audio_descriptor

The syntax of the opus_audio_descriptor is shown in Table 4 -2.

Table 4-2 opus_audio_descriptor syntax

Syntax	Number of bits	Identifier
opus_audio_descriptor() {		
descriptor_tag	**8**	**uimsbf**
descriptor_length	**8**	**uimsbf**
channel_config_code	**8**	**uimsbf**
if(channel_config_code==0x81) {		
channel_count	**8**	**uimsbf**
mapping_family	**8**	**uimsbf**
if(mapping_family>0) {		
stream_count_minus_one	**ceil(log2(channel_count))**	**uimsbf**
coupled_stream_count	**ceil(log2(stream_count+1))**	**uimsbf**
for(i=0; i<channel_count; i++) {		
channel_mapping[i]	**ceil(log2(stream_count**	**uimsbf**

} reserved } } }	+coupled_stream_count+1) N1	bsmsbf

4.4.1 Semantics for the opus_audio_descriptor

descriptor_tag: The descriptor tag is an 8-bit field which identifies each descriptor. The value of the tag for the opus_audio_descriptor is 0×??. [TODO: 0×EB appears next on the list. Can we share with DVB?]

descriptor_length: This 8-bit field specifies the total number of bytes of the data portion of the descriptor following the byte defining the value of this field.

channel_config_code: An enumeration that describes the channel configuration. The value 0×81 indicates the channel configuration is explicitly coded. All other values correspond to a particular channel configuration. Table 4 -3 gives the values for the channel_count, mapping_family, stream_count, coupled_steram_count, and channel_mapping fields for each value of channel_config_code. See below for the exact meaning of each field.

Table 4-3 channel_config_code configurations

channel_config_code	channel_count	mapping_family	stream_count	coupled_stream_count	channel_mapping
0×00	2 (dual mono)	255	1	1	{0,1}
0×01	1	0	1	0	{0}
0×02	2	0	1	1	{0,1}
0×03	3	1	2	1	{0,2,1}
0×04	4	1	2	2	{0,1,2,3}
0×05	5	1	3	2	{0,4,1,2,3}
0×06	6	1	4	2	{0,4,1,2,3,5}
0×07	7	1	4	3	{0,4,1,2,3,5,6}
0×08	8	1	5	3	{0,6,1,2,3,4,5,7}
0×09…0×7F	Reserved				
0×80	2 (dual mono)	255	2	0	{0,1}
0×81	Explicit channel configuration present				
0×82	2	1	2	0	{0,1}
0×83	3	1	3	0	{0,1,2}
0×84	4	1	4	0	{0,1,2,3}
0×85	5	1	5	0	{0,1,2,3,4}
0×86	6	1	6	0	{0,1,2,3,4,5}
0×87	7	1	7	0	{0,1,2,3,4,5,6}
0×88	8	1	8	0	{0,1,2,3,4,5,6,7}
0×89…0×FF	Reserved				

channel_count: The number of output channels. This might be different from the number of coded channels, which can change on a packet-by-packet basis. This value shall not be zero. The maximum allowable value depends on the channel mapping family. However, when using as many coded channels as output channels, it is currently not possible to store more than 250 channels in an opus_audio_descriptor, because descriptor_length is limited to 255 bytes.

mapping_family: An enumeration which defines the semantic meaning of the output channels, as defined in IETF draft-ietf-codec-oggopus [i.2]. Table 4 -4 lists the allowed channel counts and the ordered set of channel names for each mapping family. mapping_family 0 allows only a single mono or stereo stream. mapping_family 1 defines a specific set of speakers for each channel count. It is currently defined for up to 8 channels. mapping_family 255 specifies an application-defined mapping that does not provide the speaker configuration for the channels. It is used here for dual-mono streams. Values 2…254 are reserved.

Table 4-4 Channel orderings

mapping_family	channel_count	Channel Order

0	1	Mono
0	2	Left, Right
1	1	Mono
1	2	Left, Right
1	3	Left, Center, Right
1	4	Front Left, Front Right, Rear Left, Rear Right
1	5	Front Left, Front Center, Front Right, Rear Left, Rear Right
1	6	Front Left, Front Center, Front Right, Rear Left, Rear Right, LFE
1	7	Front Left, Front Center, Front Right, Side Left, Side Right, Rear Center, LFE
1	8	Front Left, Front Center, Front Right, Side Left, Side Right, Rear Left, Rear Right, LFE
255	1…255	(application defined)

stream_count_minus_one: The total number of Opus streams that make up this elementary stream, minus one. This is encoded using ceil(log2(channel_count)) bits. The actual number of Opus streams, stream_count, has the value (stream_count_minus_one+1), which can vary between 1 and channel_count. Values of stream_count larger than channel_count are not allowed.

coupled_stream_count: The number of Opus streams whose decoders should be configured to produce two channels. This is encoded using ceil(log2(stream_count+1)) bits. For example, when stream_count is 3, coupled_stream_count is encoded with 2 bits, and when stream_count is 4, coupled_stream_count is encoded with 3 bits.Values of coupled_stream_count larger than stream_count are not allowed.

channel_mapping: This is an array with one entry per output channel, indicating which coded channel should be used for each one. Each entry is encoded with M=ceil(log2(stream_count+coupled_stream_count+1)) bits. The values must be smaller than (stream_count+coupled_count), or the special value (2^M-1). If channel_mapping[i] is less than (2*coupled_count), then the output is taken from decoding stream (channel_count[i]/2) as stereo and selecting the left channel if channel_count[i] is even, and the right channel if channel_count[i] is odd. If channel_count[i] is greater than or equal to (2*coupled_count), but less than (2^M-1), then the output is taken from decoding stream (channel_count[i]-coupled_count) as mono. If channel_count[i] is (2^M-1), the corresponding output channel contains pure silence.

reserved: This field contains enough bits to pad the descriptor to a byte boundary, N1=(16-ceil(log2(channel_count))-ceil(log2(stream_count+1))+channel_count*(8-ceil(log2(stream_count+coupled_stream_count+1))))%8. An encoder shall set these bits to zero.

5 Detailed Specification for System B (DVB)

5.1 stream_type

The value of stream_type for Opus shall be 0×06 (indicating PES packets containing private data).

5.2 stream_id

The value of stream_id in the PES header shall be 0×BD (indicating private_stream_1). Multiple Opus streams may share the same value of stream_id since each stream is carried with a unique packet identifier (PID) value. The mapping of values of PID to stream_type is indicated in the transport stream Program Map Table (PMT).

5.3 opus_audio_descriptor

The syntax of the opus_audio_descriptor is shown in Table 5 -5.

Table 5-5 opus_audio_descriptor syntax

Syntax	Number of bits	Identifier
opus_audio_descriptor() {		
descriptor_tag	**8**	**uimsbf**
descriptor_length	**8**	**uimsbf**
descriptor_tag_extension	**8**	**uimsbf**
channel_config_code	**8**	**uimsbf**
if(channel_config_code==0x81) {		
channel_count	**8**	**uimsbf**
mapping_family	**8**	**uimsbf**
if(mapping_family>0) {		
stream_count_minus_one	**ceil(log2(channel_count))**	**uimsbf**
coupled_stream_count	**ceil(log2(stream_count+1))**	**uimsbf**
for(i=0; i<channel_count; i++) {		
channel_mapping[i]	**ceil(log2(stream_count +coupled_stream_count+1)**	**uimsbf**
}		
reserved	**N1**	**bsmsbf**
}		
}		
}		

5.3.1 Semantics for the opus_audio_descriptor

descriptor_tag: The descriptor tag is an 8-bit field which identifies each descriptor. The value of the tag for the opus_audio_descriptor is 0×7F, indicating an extended descriptor tag.

descriptor_length: This 8-bit field specifies the total number of bytes of the data portion of the descriptor following the byte defining the value of this field.

descriptor_tag_extension: The descriptor tag extension is an 8-bit field which expands the space of defined descriptors. The value of the tag for the opus_audio_descriptor is 0×??. [TODO: Can we share with ATSC?]

The remaining fields have the same semantics as the ATSC opus_audio_descriptor described in Section 4.4.1.

6 PES Packet Format

The first byte of a PES packet must begin a new Opus Access Unit (AU), and all PES packets must contain a whole number of AUs. The maximum duration of a single AU is equal to the maximum duration of an Opus packet, 120 ms.

6.1 opus_access_unit

An Opus AU consists of an optional control header, followed by one Opus packet for each stream specified in the channel configuration in the PMT, as described in Table 6 -6.

Table 6-6 opus_access_unit syntax

Syntax	Number of bits	Identifier

opus_access_unit() {
  if(nextbits(11)==0x3FF) {
  opus_control_header()
 	
  for(i=0; i<stream_count-1; i++) {
  self_delimited_opus_packet
  }
  undelimited_opus_packet
}		

6.1.1 Semantics for the opus_access_unit

The function nextbits() permits comparison of a bit string with the next bits to be decoded in a stream. All Opus packets within a single AU shall have the same Presentation Timestamp (PTS).

stream_count corresponds to the field in the associated opus_audio_descriptor from the PMT for this program.

opus_control_header: See Section 6.2.

self_delimited_opus_packet: A single Opus packet encoded using the self-delimited framing from Appendix B of RFC 6716 [1]. The duration of all of the Opus packets in a single AU must be equal.

6.2 opus_control_header

The opus_control_header contains optional control information for the decoder. [None of the other MPEG TS audio codecs provide sample accurate lead-in and lead-out cut points. Therefore this header is either a competitive advantage, or unnecessary cruft. It's mostly here to demonstrate how additional per-AU information could be inserted into the bitstream.]

Table 6-7 opus_access_unit syntax

Syntax	Number of bits	Identifier
opus_control_header() {		
control_header_prefix	**11**	**bslbf**
start_trim_flag	**1**	**bslbf**
end_trim_flag	**1**	**bslbf**
control_extension_flag	**1**	**bslbf**
Reserved	**2**	**bslbf**
payload_size = 0		
while(nextbits(8) == 0xFF){		
ff_byte [= 0xFF]	**8**	**uimsbf**
payload_size += 255;		
}		
payload_size_last_byte	**8**	**uimsbf**

```
    payload_size += payload_size_last_byte                    |          |
    if(start_trim_flag==1) {                                  |          |
       Reserved                                               |    3     |  bslbf
       start_trim                                             |    13    |  uimsbf
    }                                                         |          |
    if(end_trim_flag==1) {                                    |          |
       Reserved                                               |    3     |  bslbf
       end_trim                                               |    13    |  uimsbf
    }                                                         |          |
    if(control_extension_flag==1) {                           |          |
       control_extension_length                               |    8     |  uimsbf
       for(i=0; i<control_extension_length; i++) {            |          |
          reserved                                            |    8     |  bslbf
       }                                                      |          |
    }                                                         |          |
}
```

6.2.1 Semantics for the opus_control_header

control_header_prefix: The control header prefix is an 11-bit code that distinguishes it from a valid Opus packet.

start_trim_flag: A single bit that, if set, indicates the presence of a start_trim value.

end_trim_flag: A single bit that, if set, indicates the presence of an end_trim value.

control_extension_flag: A single bit that, if set, indicates the presence of extended control information.

reserved: These bits must be set to zero. [TODO: DRC, downmixing, and other metadata.]

payload_size: This shall be the total size of the Opus payload

start_trim: The number of samples per channel at 48 kHz to discard from the beginning of the Opus packets contained in this AU. This is only used at the start of a program, to compensate for padding samples inserted by the encoder. The amount the PTS advances is reduced by the corresponding amount. The number of samples cannot exceed the duration of the AU. After an AU which does not use this field to discard its entire contents, this field cannot be used again in the stream corresponding to this PID. No more than 65535 samples may be discarded in this way in total from all packets at the beginning of a stream.

end_trim: The number of samples per channel at 48 kHz to discard from the end of the Opus packets contained in this AU. This is only used at the end of a program, to allow for sample accurate durations. The amount the PTS advances is reduced by the corresponding amount. The number of samples cannot exceed the duration of the AU. No more AUs should follow an AU which contains this field. If both start_trim and end_trim are present in the same AU, then the total may not exceed the duration of the AU.

control_extension_length: The number of additional bytes in the control header.

7 T-STD Model Parameters

7.1: Transport Streams compliant with this specification shall follow the T-STD model as described in ISO/IEC 13818-1

Channels	R_{xn} (bits per second)
1-2	2000000
3-8	??
??	??
??	??

The following text is to be used when appropriate:

Proforma copyright release text block

This text box shall immediately follow after the heading of an element (i.e. clause or annex) containing a proforma or template which is intended to be copied by the user. Such an element shall always start on a new page.

Notwithstanding the provisions of the copyright clause related to the text of the present document, ETSI grants that users of the present document may freely reproduce the <proformatype> proforma in this {clause|annex} so that it can be used for its intended purposes and may further publish the completed <proformatype>.

<PAGE BREAK>

Annexes

*Each annex **shall** start on a new page (insert a page break between annexes A and B, annexes B and C, etc.).*

*Use the **Heading 8** style for the title and the Normal style for the text.*

Specify if the annex is normative or informative.

Annex <A> (normative):
Title of normative annex *(style H8)*

<Text>

Abstract Test Suite (ATS) text block

This text should be used for ATSs using either TTCN-2 or TTCN-3. In case:

- *TTCN-2 is used: attach the TTCN.MP;*
- *TTCN-3 is used: attach the TTCN-3 files and other related modules, as well as the HTML documentation of the TTCN-3 files.*

<PAGE BREAK>

Annex <X> (normative): ATS in TTCN-2 *(style H8)*

This text shall only be used for ATSs using TTCN version 2 (TTCN-2):

This ATS has been produced using the Tree and Tabular Combined Notation version 2 (TTCN-2) according to ISO/IEC 9646-3 [<x>].

<X.1> The TTCN-2 Machine Processable form (TTCN.MP) *(style H1)*

The TTCN.MP representation corresponding to this ATS is contained in an ASCII file (<any_name>.MP contained in archive <Shortfilename>.ZIP) which accompanies the present document.

<PAGE BREAK>

Annex <X+1> (normative): ATS in TTCN-3 *(style H8)*

This text shall only be used for ATSs using TTCN version 3 (TTCN-3):

This ATS has been produced using the Testing and Test Control Notation (TTCN) according to ES 201 873-1 [<x>].

Indicated here which parts of the ES 201 873 series and its versions (editions) have been used; also indicate any extensions which have been used.

<X+1.1> TTCN-3 files and other related modules *(style H1)*

The TTCN-3 and other related modules are contained in archive <Shortfilename>.zip which accompanies the present document.

<X+1.2> HTML documentation of TTCN-3 files *(style H1)*

The HTML documentation of the TTCN-3 and other related modules are contained in archive <Shortfilename>.zip which accompanies the present document.

<PAGE BREAK>

Annex <X+2> (informative):
Title of informative annex *(style H8)*

<Text>

<X+2.1> First clause of the annex *(style H1)*

<Text>

<X+2.1.1> First subdivided clause of the annex *(style H2)*

<Text>

<PAGE BREAK>

Annex <X+3> (informative): Change History

This informative annex is optional. If present, it describes the list of changes implemented in a new version of the deliverable.

Its format is tabular, it may contain the Change Request numbers and titles or textual explanations of the changes that lead to each new version number of the deliverable.

Date	Version	Information about changes
October 2011	v1.1.1	First publication of the TS after approval by TC SPAN at SPAN#19 (30 September - 2 October 2011; Prague) Rapporteur is John Smith
February 2012	v1.2.1	Implemented Change Requests: SPAN(12)20_019 Error message information clarifications SPAN(12)20_033 Revised error message information SPAN(12)20_046 update of figure 3 clause 9.2 These CRs were approved by TC SPAN#20 (3 - 5 February 2012; Sophia) Version 1.2.1 prepared by John Smith
July 2013	v1.3.1	Implemented Changes: Correction needed because the previously approved version did not contain the last version of the ASN.1 and XML attachments. Version 1.3.1 prepared by Mark Canterbury (NTAC)

<PAGE BREAK>

Annex <X+4> (informative): Bibliography

The annex entitled "Bibliography" is optional.

It shall contain a list of standards, books, articles, or other sources on a particular subject which are not mentioned in the document itself (see clause 12.2 of the EDRs http://portal.etsi.org/edithelp/Files/other/EDRs_navigator.chm).

It shall not include references mentioned in the document.

*Use the **Heading 8 style** for the title and B1+ or Normal for the text.*

- <Publication>: "<Title>".

OR

<Publication>: "<Title>".

<PAGE BREAK>

History

This clause shall be the last one in the document and list the main phases (all additional information will be removed at the publication stage).

Document history		
<Version>	<Date>	<Milestone>

A few examples:

Document history		
V1.1.1	April 2001	Publication
V1.3.1	June 2011	Pre-Processing done before TB approval e-mail: mailto:edithelp@etsi.org
V2.0.0	March 2013	Clean-up done by ***editHelp!*** e-mail: mailto:edithelp@etsi.org

Latest changes made on 2013-05-15