IMPLEMENTATION OF DTS® AUDIO IN MPEG-2 STRUCTURES AS DEFINED IN ISO/IEC 13818-1

Document No.: 9302J85300
Revision: C
Version: 1.1
Effective Date: December 2011

DTS, Inc.
5220 Las Virgenes Road
Calabasas, CA 91302 USA
www.dts.com

CONFIDENTIAL

PATENTS

NO WARRANTY

COPYRIGHT

DTS® Audio in MPEG-2 Streams

Due to ongoing improvements and revisions, DTS cannot guarantee the accuracy of printed material after date of publication nor can it accept responsibility for any errors or omissions. DTS may publish updates and revisions to this publication, however DTS has no obligation to notify you of any such update or revision and nothing herein shall be construed as creating any obligation for DTS to do so, and DTS has no obligation to update or revise this publication and nothing herein shall be construed as creating any such obligation.

Conformity with any standards contained herein shall not constitute DTS certification. No product is certified until it has passed DTS testing and DTS has issued a certification statement. Please note, products containing unreleased, beta or outdated software versions may not be certified by DTS.

The content of this publication supersedes the content of any materials previously provided by DTS pertaining to the subject matter of this publication.

CONTENTS

TABLES

1 INTRODUCTION

1.1 Document conventions

Certain stylistic conventions have been adopted for use throughout this document as a matter of convenience and readability. Some of those conventions are described here.

Normative references, as listed in Section 1.2, are indicated by their bracketed number value on the first cited occurrence for assignment. For example, "ISO/IEC 13818-1 [1]".

Hexadecimal numerical values are indicated as '*n*h', or '0x*n*'.

Binary numerical values are indicated as '*n*b', or '0b*n*'.

Character arrays as parameters are enclosed in a single quotation, such as 'abcd'.

Normative words SHALL and SHALL NOT indicate required applications. Optional or descriptive language also exists to denote suggested (should) or permissive (may) applications or methods of implementation. All bit fields, indicated using the bit(x) notation, applying enumerated sequence as parameter values are unsigned most significant bit first bit fields or integers. If any of the possible enumerated values are not defined in this document, they shall be considered reserved for future use.

Parameters, data fields, or enumerated values marked or implied as 'reserved' may be defined in future or other specifications, therefore they shall not be read by any device only applying this specification. Any data structures that are created using only this specification shall set reserved fields to 0.

1.2 Normative References

The following documents are essential in applying the specification described here-in.

[1] ITU-T Recommendation H.222.0 / ISO/IEC 13818-1, "Information Technology – Generic coding of moving pictures and associated audio information: Systems", International Standards Organization, www.iso.ch; International Electrotechnical Commission, www.iec.ch

[2] ETSI EN 300 468, Version 1.11.1, "Digital Video Broadcast (DVB); Specification for Service Information (SI) in DVB Systems", April 2010, European Telecommunication Standards Institute, www.etsi.org

[3] ETSI EN 101 154, Version 1.9.1, "Digital Video Broadcast (DVB); Specification for Video and Audio Coding in Broadcasting Applications based on the MPEG-2 Transport Stream", September 2009, European Telecommunication Standards Institute, www.etsi.org

[4] ISO/IEC 639-3, "Codes for the representation of names of language – Part 3: Alpha-3 code for comprehensive coverage of languages", International Standards Organization, www.iso.ch; International Electrotechnical Commission, www.iec.ch

1.3 Informative References

The following referenced documents are not essential to the use of the present document but they assist the user with regard to a particular subject area. For all references, the latest version of the referenced document applies.

[5] "DTS Coherent Acoustics Core", *(DTS Document No.: 9302F33500)*

[6] "DTS-HD Substream Decoder Interface", *(DTS Document No.: 9302F30400)*

[7] "DTS Coherent Acoustics Extensions", *(DTS Document No.: 9302F41300)*

[8] "DTS-HD PBR API Library Description", *(DTS Document No.: 9302J81100)*

1.4 Terms, Definitions, and Acronyms

The following terms, definitions and abbreviations shall apply to all clauses in this document.

AU: access unit, the data representing an autonomous segment of coded audio or one audio frame.

audio stream: A sequence of synchronized audio frames

audio frame: A component of a substream that corresponds to a certain number of PCM audio samples.

CBR: Constant Bit Rate

core substream: A DTS audio stream, or a component of a DTS audio stream, conforming to [5] and always begins with the 32-bit Sync word of 0x7FFE8001.

duration: The time represented by one decoded audio frame, may be represented in audio samples per channel at a specific audio sampling frequency or in seconds.

elementary stream (ES): A generic term for one of the coded bitstreams carried in a sequence of PES packets, as described in [1].

extension substream: A DTS audio stream, or a component of a DTS audio stream, conforming to [6] and always begins with the 32-bit sync word of 0x64582025.

dependant substream: An extension substream that is associated with a core substream.

frame: audio frame

LFE: Low Frequency Effects or subwoofer channel

lsb: least significant bit

PES: Packetized Elementary Stream

PES payload: The portion of the PES packet following the PES header.

PMT: Program Map Table, as described in [1].

presentation time-stamp (PTS): A field that may be present in a PES packet header that indicates time that a presentation unit is presented in the systems target decoder, as described in [1]

VBR: Variable Bit Rate

XCH: A DTS core extension that adds a center surround channel.

XLL: The DTS-HD lossless audio coding extension, a logical element within the DTS elementary stream containing compressed audio data that will decode into bit exact representation of the original signal.

XXCH: Channel extension. The XXCH extension can add up to 32 channels at arbitrary locations to a DTS core (which is limited to up to 5.1 channels) in either the core substream, or in an extension substream.

1.5 Referenced Parameters

A number of parameters referenced in this specification are defined in other documents. These parameters are listed below in Table 1-1.

Table 1-1: Cross reference to parameters

Parameter	Reference [see 1.2 Normative References]
descriptor_tag	ETSI EN300 468 [2]
descriptor_tag_extension	ETSI EN300 468
BS_n	ISO/IEC 13818-1 [1]
Data_Alignment_Indicator	ISO/IEC 13818-1
ES_info_length	ISO/IEC 13818-1
format_identifier	ISO/IEC 13818-1
PTS	ISO/IEC 13818-1
private_stream_1	ISO/IEC 13818-1
Rx_n	ISO/IEC 13818-1
stream_id	ISO/IEC 13818-1
TS_program_map_section	ISO/IEC 13818-1

2 APPLICATION OF DTS BITSTREAMS TO MPEG-2 STREAMS

2.1 Introduction

Two related generations of DTS formats exist, the original DTS core format, and the expanded DTS-HD format. The features of DTS-HD are a superset of the original DTS features and, in many cases, a direct extension to the original format. The original DTS format is referred to as the "core substream", or may be referred to as "the core". The new component added for DTS-HD is called the "extension substream", or just "the extension". The extension, when it exists with the core, may serve to enhance the audio stored in the core substream, as is the case with most DTS-HD 5.1 audio streams. The extension substream may also add new information that does not exist in the core, as is the case with most DTS-HD 7.1 audio streams. The extension may also exist without the core. An example of this application would be DTS Express™. In order to manage the existence of the new extension substream and the various new coding profiles an additional format identifier and audio stream descriptor is necessary. This new structures can accommodate core only formats as well as extension only and core + extension combinations.

2.2 Buffering Model

The DTS buffering model is designed in accordance with ISO/IEC 13818-1. Refer to the derivation of BS_n for audio elementary streams.

- For DTS core streams, the main audio buffer size (BS_n) has a fixed value of 9088 bytes, with a drain rate (Rx_n) of 2 Mbps. The fixed value above (9088 bytes) was calculated from a double buffer (2*4096 bytes) plus jitter (384 bytes) + packet bursts (512 bytes).
- For DTS-HD Lossless formats, the value of BS_n shall have a fixed value of 66 432 bytes, with an Rx_n value of 32 Mbps.
- For all other DTS-HD formats, the value of BS_n shall have a fixed value of 17 814 bytes, with an Rx_n value of 8 Mbps.

2.3 Signaling

2.3.1 PSI Signaling in the PMT

2.3.1.1 Stream Type

All DTS and DTS-HD elementary streams SHALL use a stream_type = 0x06, indicating private data, in accordance with ITU-T Recommendation H.222.0 / ISO/IEC 13818-1 [1].

2.3.1.2 Registration Descriptor

In a Transport Stream environment, a registration descriptor with a registered format identifier is required to identify the DTS formats. See Table 2-1 for a list of codes to be used. These assignments are consistent with the definitions in ETSI TS 101 154 [3]

Table 2-1: DTS Format Identifiers

label	format_identifier	audio frame duration (samples)
'DTS1'	0x44545331	512
'DTS2'	0x44545332	1024
'DTS3'	0x44545333	2048
'DTSH'	0x44545348	various

For the original DTS formats, (or core substream only), unique format_identifier values have been assigned in association with audio frame duration as shown in Table 2-1. The use of these format identifiers SHALL be consistent with the nblks field in DTS_audio_stream_descriptor() whose relationship is as follows:

```
audio frame duration (samples) = (nblks+1)*32
```

When DTS-HD_audio_stream_descriptor() is used, the format identifier 'DTSH' SHALL be used in the registration descriptor.

The registration descriptor SHALL be included in the descriptor loop of the PMT describing the elementary stream immediately following the ES_info_length field.

2.3.1.3 DTS Audio Stream Descriptor

The DTS audio stream descriptor is associated with format_identifiers DTS1, DTS2, and DTS3. This descriptor may be used when only a core substream is present. In this case, the DTS audio stream descriptor SHALL be included in the TS_program_map_section descriptor loop immediately following the DTS registration descriptor.

The DTS audio stream descriptor is described in the structure defined below.

```
DTS_audio_stream_descriptor() {
    bit(8)    descriptor_tag;
    bit(8)    descriptor_length;
    bit(4)    sample_rate_code;
    bit(6)    bit_rate_code;
    bit(7)    nblks;
    bit(14)   fsize;
    bit(6)    surround_mode;
    bit(1)    lfe_flag;
    bit(2)    extended_surround_flag;
    bit(8)    component_type;
    for (i=0; i<N; i++)
        bit(8)    additional_info_byte[i];
}
```

where:

descriptor_tag: this value is defined in ETSI EN300 468 for DVB systems, and is registered with a value of 0x7B. Other systems may associate a different value with DTS audio.

descriptor_length: this 8-bit field specifies the total number of bytes following the descriptor_length field. It SHALL be used to determine the number of additional_info_bytes contained in the audio_stream_descriptor.

sample_rate_code: is described in Table 2-2.

Table 2-2: sample_rate_code

sample_rate_code	Sampling Frequency
1	8 kHz
2	16 kHz
3	32 kHz
4	64 kHz
6	11.025 kHz
7	22.050 kHz
8	44.1 kHz
9	88.2 kHz
11	12 kHz
12	24 kHz
13	48 kHz
14	96 kHz

bit_rate_code: indicated the approximate bit rate of the DTS elementary stream, as shown in Table 2-3.

Table 2-3: bit_rate_code

Transmission Bit Rate (kbits/s)	bit_rate_code					
	b5	b4	b3	b2	b1	b0
128	x	0	0	1	0	1
192	x	0	0	1	1	0
224	x	0	0	1	1	1
256	x	0	1	0	0	0
320	x	0	1	0	0	1
384	x	0	1	0	1	0
448	x	0	1	0	1	1
512	x	0	1	1	0	0
576	x	0	1	1	0	1
640	x	0	1	1	1	0
768	x	0	1	1	1	1
960	x	1	0	0	0	0
1024	x	1	0	0	0	1
1152	x	1	0	0	1	0
1280	x	1	0	0	1	1
1344	x	1	0	1	0	0
1408	x	1	0	1	0	1
1411.2	x	1	0	1	1	0
1472	x	1	0	1	1	1
1536	x	1	1	0	0	0
open	x	1	1	1	0	1

- Note: the ‘x’ in b5 under bit_rate_code indicates that the bit is reserved and, for the purposes of this definition, shall be ignored

nblks: indicates the number of PCM Sample Blocks per frame. The valid range of nblks is 5 to 127. The total number of PCM sample blocks is equal to nblks+1. A block = 32 PCM samples per channel. For normal frames, this indicates a window size of either 2 048 (nblks = 63), 1 024 (nblks = 31), or 512 (nblks = 15). For termination frames nblks can be any value in its valid range.

fsize: indicates the number of bytes in the current audio frame. The total number of bytes in the current frame of the core substream is fsize + 1. The valid range for fsize is 95 to 8192.

surround_mode: indicates the core substream channel configuration, not including any additional extensions, as shown in Table 2-4.

Table 2-4: surround_mode

surround_mode	Channels (substream_core.channel_count)	Layout	Description
0	1	1/0	mono
2	2	2/0	L + R (stereo)
3	2	2/0	(L+R) + (L-R) (sum-difference)
4	2	2/0	LT +RT (left and right total)
5	3	3/0	C + L + R
6	3	2/1	L + R+ S
8	4	2/2	L + R+ SL+SR
9	5	3/2	C + L + R+ SL+SR
10 to 31			User defined
NOTE: L =left, R = right, C =centre, SL = surround left, SR = surround right, T = total.			

lfe_flag: indicates the presence of an LFE channel, as described in Table 2-5.

Table 2-5: lfe_flag

lfe_flag	description
0	LFE is not present
1	LFE is present

extended_surround_flag: indicates whether an extended surround channel intended for the rear center position exists, and some information about how it was generated, as shown in Table 2-6.

Table 2-6: extended_surround_flag

extended_surround_flag	Description
0	No Extended Surround
1	Matrixed Extended Surround
1	Matrixed Extended Surround with Fs≥ 48 kHz
2	Discrete Extended Surround
3	Invalid

component_type: is described in section 2.3.2.1.

additional_info_byte: This byte array of from 0 to N bytes is reserved for future use. The number of bytes in this array is determined by comparing descriptor_length to the number of bytes parsed up to and including component_type.

2.3.1.4 DTS-HD Audio Stream Descriptor

The DTS-HD audio stream descriptor is associated with the DTSH format identifier. This format SHALL be used for all audio streams consisting of a core and extension, and streams containing only an extension substream. Since DTS audio is a subset of DTS-HD, DTS formats may also use the DTS-HD descriptor (i.e. the case of core substream without an extension substream). Implementations capable of parsing DTS-HD streams SHALL accept both methods of signaling DTS core audio.

The DTS-HD audio stream descriptor SHALL be included in the TS_program_map_section descriptor loop immediately following the DTS registration descriptor.

- Note that the core substream descriptor values are always valid for the core as an independent asset in the event that a receiver will only decode the core substream.

The DTS-HD audio stream descriptor is described below:

```
DTS-HD_audio_stream_descriptor() {
    bit(8)    descriptor_tag;                    // extension_descriptor = 0x7F if DVB
    bit(8)    descriptor_length;
    if (DVB System)
        bit(8)    descriptor_tag_extension;      // if DVB: = 0x0e . Field absent if not DVB
    bit(1)    substream_core_flag;
    bit(1)    substream_0_flag;
    bit(1)    substream_1_flag;
    bit(1)    substream_2_flag;
    bit(1)    substream_3_flag;
    bit(3)    reserved;                          //replaced variable pad
    if (core_substream_flag)
        struct substream substream_core;
    if (substream_0_flag)
        struct substream substream_0;
    if (substream_1_flag)
        struct substream substream_1;
    if (substream_2_flag)
        struct substream substream_2;
    if (substream_3_flag)
        struct substream substream_3;
for (i=0; i<N; i++)
    bit(8)    additional_info_byte[i];
}
```

```
struct substream {
    bit(8)    substream_length
    bit(3)    num_assets;
    bit(5)    channel_count;
    bit(1)    LFE_flag;
    bit(4)    sampling_frequency;
    bit(1)    sample_resolution;
    bit(2)    reserved;
    for (i=0; i ≤ num_assets; i++)
        struct s_asset asset [i];
}

struct s_asset {
    bit (5)   asset_construction;
    bit(1)    vbr_flag;
    bit(1)    post_encode_br_scaling_flag;
    bit(1)    component_type_flag;
    bit(1)    language_code_flag;
    if (post_encode_br_scaling_flag)
        bit(13)   bit_rate_scaled;
    else
        bit (13)  bit_rate;
    bit(2)    reserved;
    if (component_type_flag)
        bit(8)    component_type;
    if (language_code_flag)
        bit(24)   ISO_639_language_code;
}
```

where:

descriptor_tag: For DVB systems this value is set to 0x7F indicating the extended descriptor is in use as defined in ETSI EN300 468. For other implementations this value may be set to a value chosen by the system to indicate DTS-HD audio.

descriptor_length: this 8-bit field specifies the total number of bytes following the descriptor_length field. It SHALL also be used to determine the number of additional_info_bytes contained in the audio stream descriptor.

descriptor_tag_extension: If the extended descriptor is in use in a DVB system, this 8-bit field is set to 0x0e to indicate DTS-HD audio, and will be defined in a future version of ETSI EN 300 468. If the extended descriptor is not in use, this field is not present.

substream_core_flag: SHALL be set to 1 if a core substream exists in the audio stream.

substream_0_flag: SHALL be set to 1 if an extension substream with nuExtSSIndex =0 exists in the audio stream.

substream_1_flag: SHALL be set to 1 if an independent extension substream with nuExtSSIndex =1 exists in the audio stream.

substream_2_flag: SHALL be set to 1 if an independent extension substream with nuExtSSIndex =2 exists in the audio stream.

substream_3_flag: SHALL be set to 1 if an independent extension substream with nuExtSSIndex =3 exists in the audio stream.

reserved: The reserved bits throughout the various sub-structures serve to force byte alignment of several major subcomponents in the DTS-HD descriptor. These bits are reserved for future definition and shall be ignored by receivers built to this version of specification should they become defined in the future.

substream_length: this 8-bit field specifies the total number of bytes following the substream_length field in the substream structure, including the embedded asset structures.

additional_info_byte: This byte array of from 0 to N bytes is reserved for future use. The number of bytes in this array is determined by comparing descriptor_length to the number of bytes parsed up to the end of the last substream element.

num_assets: represents the number of audio assets stored in the substream. The number of audio assets stored in the substream is equal to num_assets+1. For substream_core num_assets is always 0. For all independent extension substreams, this value is identical to nuNumAssets in the extension substream header.

channel_count: The maximum number of output channels including LFE (if present). Note that channel_count represents the maximum number of channels after all relevant assets are mixed together in multi-asset presentations and may be less than or equal to the number of channels of all assets combined.

lfe_flag: If lfe_flag is set to 1, then this substream contains an LFE channel.

sampling_frequency: Use Table 2-7 to determine the maximum sampling frequency stored in the elementary stream. Note that not all values in the table are valid for the substream_core parameter.

Table 2-7: sampling_frequency

substream_core.sampling_frequency substream_*n*.sampling_frequency	Sampling Frequency
0	8 kHz
1	16 kHz
2	32 kHz
3	64 kHz
4*	128 kHz
5	22.05 kHz
6	44.1 kHz
7	88.2 kHz
8*	176.4 kHz
9*	352.8 kHz
10	12 kHz
11	24 kHz
12	48 kHz
13	96 kHz
14*	192 kHz
15*	384 kHz

- Note: Values indicated with (*) are invalid for substream_core.sampling_frequency

sample_resolution: This parameter indicates whether the decoded audio should be treated as 16-bit or 24-bit samples. If any resolution ≤ 16 bits are indicated, sample_resolution = 0, indicating 16-bit audio. Otherwise, sample_resolution = 1, indicating 24-bit audio.

asset_construction: This parameter provides details about the internal construction of the audio stream. and is interpreted according to Table 2-8.

Table 2-8: Interpreting asset_construction

asset_construction	Core substream				Extension substream					
	Core	XCH	X96	XXCH	Core	XXCH	X96	XBR	XLL	LBR
1	✓									
2	✓	✓								
3	✓			✓						
4	✓		✓							
5	✓					✓				
6	✓							✓		
7	✓	✓						✓		
8	✓			✓				✓		
9	✓					✓		✓		
10	✓						✓			
11	✓	✓					✓			
12	✓			✓			✓			
13	✓					✓	✓			
14	✓								✓	
15	✓	✓							✓	
16	✓		✓						✓	
17									✓	
18										✓
19					✓					
20					✓	✓				
21					✓				✓	

vbr_flag: This flag is set to 1 if the audio asset has a variable bit rate, otherwise this flag is 0.

post_encode_br_scaling_flag: This flag is set to 1 if scaling of the bit stream has occurred after it was encoded, otherwise this flag is 0.

component_type_flag: SHALL be set to 1 if the field component_type is present.

language_code_flag: SHALL be set to 1 when ISO_639_language_code field is present.

bit_rate_scaled: represents the scaled bit rate of the coded elementary stream as a 10.3 unsigned fractional fixed point value. This field is exists in the descriptor when post_encode_br_scaling_flag = 1.

If the stream is variable bit rate, and dynamically updating the bit rate field is not possible or practical due to system limitations, or exceeds 8191 kbits/sec, then bit_rate_scaled SHALL be set to 0.

bit_rate: is a 13-bit unsigned integer representing the bit rate of the coded elementary stream in kbits/s, ranging from 1 to 8191 kbits/s. This field exists in the descriptor when post_encode_br_scaling_flag = 0.

If the stream is variable bit rate, and dynamically updating the bit rate field is not possible or practical due to system limitations, then bit_rate SHALL be set to 0.

component_type: is described in Section 2.3.2.1.

language_code_flag: This flag SHALL be set to 1 when ISO_639_language_code field is present.

I**SO_639_language_code**: This 24-bit language code conforms to the ASCII language codes described in ISO/IEC 639 [4].

2.3.2 DVB-SI Signaling

2.3.2.1 Component Descriptor

Use of the component descriptor is optional.

```
component_descriptor() {
    bit(8)    descriptor_tag;
    bit(8)    descriptor_length;
    bit(4)    reserved
    bit(4)    stream_content
    bit(8)    component_type
    bit(8)    component_tag
    bit(24)   ISO_639_language_code
    for (i=0; i<N; i++){
        bit(8)    text_char[i]
    }
}
```

stream_content: This 4-bit field specifies the type of stream. The coding of this field as 0x07 is reserved for DTS in ETSI EN 300 468.

component_type: This 8-bit field specifies the type of the audio component. Component types 0x00 to 0x7F are reserved for DTS in ETSI EN 300 468, Chapter 6.2.8, table 26, and is defined in Table 2-9.

Table 2-9: component_type

component_type bits	Description
b7 (MSB)	reserved
b6	full service flags (see Table 2-10)
b5 to b3	service type flags (see Table 2-11)
b2 to b0	number of channels flags (see Table 2-12)

Table 2-10: full service flag

full service flag (b6)	Description
0	Decoded audio stream is intended to be combined with another decoded audio stream before presentation
1	Decoded audio stream is a full service (suitable for decoding and presentation to the listener)

Table 2-11: Service Type flags

Service Type Flags			Description	Restrictions	
b5	b4	b3		full service flag (b6)	number of channel flags (b2 to b0)
0	0	0	Complete Main (CM)	must be set to 1	
0	0	1	Music and Effects (ME)	must be set to 0	
0	1	0	Visually Impaired (VI)		
0	1	1	Hearing Impaired (HI)		
1	0	0	Dialogue (D)	must be set to 0	
1	0	1	Commentary (C)		must be set to 000
1	1	0	Emergency (E)	must be set to 1	must be set to 000
1	1	1	Voiceover (VO)	must be set to 0	must be set to 000
1	1	1	Reserved	must be set to 1	
Note: The values of the service type flags shall only be considered valid if the conditions identified in the restrictions column are satisfied					

Table 2-12: Number of Channels flags

number of channels flags			Description
b2	b1	b0	
0	0	0	Mono
0	0	1	Reserved
0	1	0	2 channel (stereo, LoRo)
0	1	1	2 channel matrix encoded (stereo, LtRt)
1	0	0	Multichannel audio (>2 channels)
1	0	1	Reserved
1	1	0	Reserved
1	1	1	Reserved

2.4 Elementary Stream Encapsulation

2.4.1 Stream ID

All DTS and DTS-HD elementary streams SHALL use a stream_id = 0xBD, indicating private stream 1, in accordance with ITU-T Recommendation H.222.0 / ISO/IEC 13818-1 [1]. Multiple DTS / DTS-HD streams may share the same value of stream_id since each stream is carried with a unique PID value. The mapping of values of PID to stream_type is indicated in the transport stream PMT.

2.4.2 Audio Access Unit Alignment in the PES packet

A valid sync word SHALL be aligned with the start of the PES packet data area. Valid DTS sync words are listed in Table 2-13. Data_Alignment_Indicator in the PES packet header SHALL indicate sync word alignment.

Table 2-13: DTS-HD Sync Words

name	sync word	description
DTS_SYNCWORD_CORE	0x7ffe8001	core substream
DTS_SYNCWORD_SUBSTREAM	0x64582025	extension substream

When a core substream is present, DTS_SYNCWORD_CORE SHALL be aligned to the beginning of the PES payload. When only an extension substream is present, DTS_SYNCWORD_SUBSTREAM SHALL be aligned to the beginning of the PES payload.

A PES packet of DTS audio SHALL contain at least one complete audio access unit. Multiple complete access units are permitted in a PES packet only when the ES consists of a single substream.

The DTS audio descriptor parameter fsize (see section 2.3.1.3) and the DTS-HD audio descriptor parameter substream_length (see section 2.3.1.4) describe the number of bytes in the core substream and extension substream (respectively).

If multiple substreams are present, the access units shall maintain an interleaved order of presentation, as illustrated below in Figure 2.1.

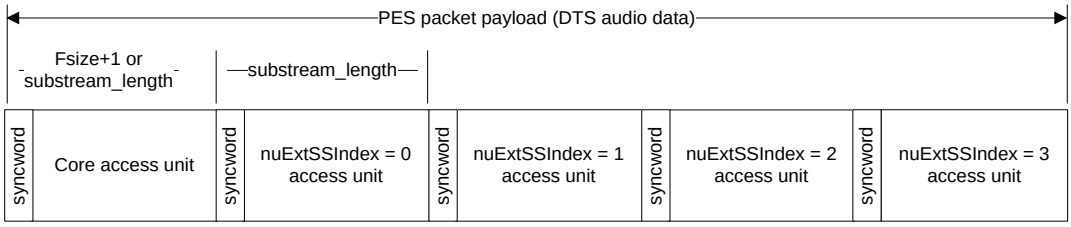

Figure 2.1: PES packet payload

3 SIGNALING FOR DTS NEURAL SURROUND™

3.1 Introduction

DTS Neural Surround constitutes a method of encoding and decoding additional audio channels into a host audio stream. A common application example would be to encode 5.1 channel surround sound into a stereo audio program.

DTS Neural Surround coding is an active intensity phase encoding methodology, and does not carry embedded metadata signaling. A system unaware of this processing will operate without impediment. The nature of this encoding permits the carriage independent of audio encoding algorithm, provided sufficient quality is maintained.

A common system application example would be to use DTS Neural Surround to encode 5.1 channel surround sound for a stereo MPEG 1 layer 2 host audio, enabling the delivery of surround sound audio in early generations of digital media distribution systems designed for stereo audio distribution.

3.2 Signaling

3.2.1 PSI Signaling in the PMT

The DTS_Neural() descriptor SHALL follow the associated audio descriptor in the descriptor loop of the PMT.

3.2.2 DTS Neural Descriptor

The DTS Neural descriptor is described in the structure s follows:

```
DTS_Neural_descriptor(){
    bit(8)    descriptor_tag;
    bit(8)    descriptor_length;
    if (DVB)
        bit(8)    descriptor_tag_extension;
    bit(8)    config_ID;
    for (i=0; i<N; i++)
        bit(8)    additional_info_byte[i];
}
```

Where:

descriptor_tag: For DVB systems, this value is set to 0x7F indicating the extended descriptor is in use as defined in ETSI 300 468. For other implementations this value may be set to a value chosen by the system to indicate DTS Neural Surround encoding.

descriptor_length: this 8-bit field specifies the total number of bytes following the descriptor_length field. It SHALL also be used to determine the number of additional_info_bytes contained in the audio stream descriptor.

descriptor_tag_extension: If the extended descriptor is in use in a DVB system, this 8-bit field is set to 0x0f to indicate DTS Neural Surround encoding, and will be defined in a future version of ETSI EN 300 468. If the extended descriptor is not in use, this field is not present.

config_ID: Dependant on the audio channel configuration of the host audio stream.

For a stereo host audio stream:

Table 3-1: Neural Stereo Host

config_ID	Original audio configuration	Original channel count*
0	Unknown or undefined	
1	L, R	2
2	L, R, C	3
3	L, R, Ls, Rs	4
4	L, R, C, Ls, Rs	5
5	L, R, C, Ls, Rs, Cs	6
6	L, R, C, Ls, Rs, Lb, Rb	7
7	L, R, Ls, Rs, Cs	5
8	L, R, Ls, Rs, Lb, Rb	6

For a 5.1 host audio stream:

Table 3-2: Neural 5.1 Host

config_ID	Original audio configuration	Original channel count
0	Unknown or undefined	
1	L, R, C, LFE, Ls, Rs	5.1
2	L, R, C, LFE, Ls, Rs, Cs	6.1
3	L, R, C, LFE, Ls, Rs, Lb, Rb	7.1

*Note that the LFE channel is omitted from the original audio configuration for stereo host audio streams. If an LFE channel existed in the original audio configuration, that channel was mixed into the full band channels during DTS Neural Surround encoding. An LFE channel can be generated by a bass management system.

additional_info_byte: This byte array of from 0 to N bytes is reserved for future use. The number of bytes in this array is determined by comparing descriptor_length to the number of bytes parsed up to and including config_ID.

Annex A Application Guidelines: Determining Descriptor Parameters from the Elementary Stream

A.1 Introduction

The information needed to derive the DTS and DTS-HD audio stream descriptor parameters may be extracted from the respective elementary stream. DTS has tools available to implementers that will analyze DTS elementary streams and extract the information necessary to populate the parameters in the dts and dts-hd descriptors. The associated document, DTS document number: 9302J81100 [8], describes the function calls and return structures. To obtain this tool and additional documentation, please direct all document requests to DTS Licensing at LicensingAdministration@dts.com

Additionally, the following sections will assist those that prefer to build their own stream parser.

A.2 Referenced Parameters

A number of parameters referenced in this specification are defined in other documents. These parameters are listed below in Table A 1.

Table A 1 : Cross-reference to Parameters used in Annex A

AMODE	DTS Core [5]
CHS	DTS Core
EXT_AUDIO	DTS Core
EXT_AUDIO_ID	DTS Core
FSIZE	DTS Core
LFF	DTS Core
NBLKS	DTS Core
PCMR	DTS Core
RATE	DTS Core
SFREQ	DTS Core
XXChSpkrLayoutMask	DTS Extensions [7]
nExtSSIndex	DTS-HD Substream [6]
nuExSSFrameDurationCode	DTS-HD Substream
nuAssetDescriptFsize	DTS-HD Substream
nuAssetFsize	DTS-HD Substream
nuBitResolution	DTS-HD Substream
nuCodingMode	DTS-HD Substream
nuCoreExtensionMask	DTS-HD Substream
nuCoreSpkrActivityMask	DTS-HD Substream
nuExtSSFsize	DTS-HD Substream
nuMaxSamplerate	DTS-HD Substream
nuNumAssets	DTS-HD Substream
nuRefClockCode	DTS-HD Substream
nuSpkrActivityMask	DTS-HD Substream
RefClockPeriod	DTS-HD Substream

A.3 DTS Audio Stream Descriptor

sample_rate_code indicates the maximum sampling frequency of the audio stored in the elementary stream. Normally SFREQ in the core substream header will directly correlate to sample_rate_code. The exception to this is when the X96 extension is present. SFREQ will continue to represent either 44.1 kHz or 48 kHz when X96 is present. In this case sample_rate_code = SFREQ + 1. The presence of X96 can be determined by EXT_AUDIO_ID. If EXT_AUDIO_ID = 2, then X96 is present.

bit_rate_code This parameter is exactly the same as RATE in the DTS core header.

nblks is identical to NBLKS in the DTS core substream header.

fsize is identical to FSIZE in the core substream header.

surround_mode is identical to AMODE in the core substream header.

lfe_flag may be determined by LFF in the core substream header. If LFF is set to 1 or 2, then an LFE channel is present and lfe_flag should be set to 1, otherwise it should be cleared to 0.

extended_surround_flag may be determined by using several parameters from, the core substream header, as indicated in Table A 2.

Table A 2 : extended_surround_flag detail

EXT_AUDIO	EXT_AUDIO_ID	PCMR (lsb)	extended_surround_flag	Description
0	0	x	0	No Extended Surround
0	0	1	1	Matrixed Extended Surround
1	2	1	1	Matrixed Extended Surround with Fs≥ 48 kHz
1	0	x	2	Discrete Extended Surround
all other combinations			3	Undefined; These streams SHALL use "DTSH" as their DTS Format Identifier

Note that the extended_surrpund_flag setting of 3 is invalid. Elementary streams whose parameters do not conform to one of the defined setting must use the DTS-HD Audio Descriptor.

A.4 DTS-HD Audio Stream Descriptor

num_assets for any extension is identical to nuNumAssets in the extension substream header.

substream_core.channel_count can be calculated by using Table 2-4 to determine the core configuration, then accounting for the presence of whether an LFE channel exists and finally accounting for the presence of additional surround channels. If an XXCH extension is present in the core substream, (see Table 2-8, asset_construction = 3), then CoreSpkrActivityMask and XXChSpkrLayoutMask can be used to determine this value.

For **substream_*n*.channel_count**, usually the extension substream header parameters nuSpkrActivityMask or, in the case of mixed assets nuMixOutChMask can be use to determine the maximum number of output channels. One exception is the case of LtRt stereo streams, where the channel count is 2.

substream_core.lfe_flag may be determined by LFF in the core substream header. If LFF is set to 1 or 2, then an LFE channel is present and substream_core.lfe_flag should be set to 1 otherwise it should be cleared to 0.

substream_*n*. lfe_flag is indicated in nuSpkrActivityMask.

substream_core.sampling_frequency can be determined by a combination of parameters in the core substream header, as indicated in Table A 3.

Table A 3 : Setting substream_core.sampling_frequency

EXT_AUDIO_ID	SFREQ	substream_core.sampling_frequency	Sampling Frequency
≠2	1	0	8 kHz
	2	1	16 kHz
	3	2	32 kHz
2	3	3	64 kHz
≠2	7	5	22.05 kHz
	8	6	44.1 kHz
2	8	7	88.2 kHz
≠2	11	10	12 kHz
	12	11	24 kHz
	13	12	48 kHz
2	13	13	96 kHz

substream_*n*.sampling_frequency is identical to nuMaxSamplerate in the extension substream header.

substream_core.sample_resolution is indicated by PCMR in the core substream header.
substream_*n*.sample_resolution is indicated by nuBitResolution .

asset_construction can be using the elementary stream header parameters as constructed from Table A 2, shown below.

Table A 4 : Determining asset_construction from the elementary stream

core substream header		extension substream header		asset_construction
EXT_AUDIO	EXT_AUDIO_ID	nuCodingMode	nuCoreExtensionMask	
0	0	Not Applicable		1
1	0			2
1	6			3
1	2			4

core substream header		extension substream header		asset_construction
EXT_AUDIO	EXT_AUDIO_ID	nuCodingMode	nuCoreExtensionMask	
Not Applicable		0	0x041	5
		0	0x021	6
		0	0x029	7
		0	0x023	8
		0	0x061	9
		0	0x081	10
		0	0x089	11
		0	0x083	12
		0	0x0C1	13
		0	0x201	14
		0	0x209	15
		0	0x205	16
		1	Don't care	17
		2		18
		0	0x010	19
		0	0x050	20
		0	0x210	21

bit_rate_scaled can be calculated as:

```
substream_n.asset[i].bit_rate =
                    float2fixed((nuAssetDescriptFsize + nuAssetFsize) / Frame duration / 1000)
```

where

```
uint13 float2fixed(float f_value) //convert floating point value to fixed point (10.3) number
{
    unsigned int i_value
    i_value = (unsigned int)(f_value * 8)   // account for the 3 fractional bits
    if (i_value < 8192)                     // maximum value that can be represented is 1023.875
        return (unint13) (i_value)
    else
        return (0)
}
```

Please see section A.5 for calculation of the frame duration.

bit_rate can be calculated as:

```
substream_core.asset[0].bit_rate = FSIZE * 8 / Frame duration in seconds / 1000

substream_n.asset[i].bit_rate =
                    (nuAssetDescriptFsize + nuAssetFsize) * 8 / Frame duration in seconds / 1000
```

Please see section A.5 for calculation of the frame duration.

A.5 Calculating Frame Duration

Frame duration is the elapsed playing time of one audio frame, which is also one audio access unit. The time duration of one audio access unit can be calculated by dividing the audio frame duration in samples by the audio sampling frequency.

A.5.1 Frame Duration from Core Substream Metadata

In the case of a core substream, the audio frame duration for a normal (non-termination) frame is nblks and the audio sampling frequency is determined from Table 2-2. Thus the frame duration in seconds is:

frame duration(seconds) = (nblks + 1) * 32 / *Audio Sampling Frequency*

A.5.2 Frame Duration from Extension Substream Metadata

In the cases where an extension substream is present, the parameters nuRefClockCode and nuExSSFrameDurationCode are used to determine the audio frame duration.

The frame duration is expressed by the number of clock cycles using the reference clock indicated by the value in RefClockPeriod. The number of clock cycles is derived from nuExSSFrameDurationCode by multiplying its value by 512. The actual duration in seconds is calculated in the following manner:

*frame duration(seconds) = nuExSSFrameDurationCode * 512 * RefClockPeriod*

Table A 5 : Reference Clock Period

nuRefClockCode	RefClockPeriod [seconds]
0	1.0 / 32000.0
1	1.0 / 44100.0
2	1.0 / 48000.0
3	Unused