UNIVERSAL SERIAL BUS
DEVICE CLASS DEFINITION
FOR
AUDIO DATA FORMATS

Release 3.0

September 22, 2016

SCOPE OF THIS RELEASE

This document is the Release 3.0 of this device class definition.

CONTRIBUTORS

Joe Scanlon	Advanced Micro Devices
Rhoads Hollowell	Apple Inc.
Girault Jones	Apple Inc.
Matthew X. Mora	Apple Inc.
Tzung-Dar Tsai	C-Media Electronics, Inc.
Brad Lambert	Cirrus Logic, Inc.
Dan Bogard	Conexant Systems, Inc.
Pete Burgers	DisplayLink (UK), Ltd.
David Roh	Dolby Laboratories, Inc.
Leng Ooi	Google, Inc.
Pierre-Louis Bossart	Intel Corporation
David Hines	Intel Corporation
Abdul Rahman Ismail (Co-Chair)	Intel Corporation
Devon Worrell	Intel Corporation
Chandrashekhar Rao	Logitech, Inc.
Terry Moore	MCCI Corporation
Alex Lin	MediaTek, Inc.
Bala Sivakumar	Microsoft Corporation
Geert Knapen (Co-Chair & Editor)	NXP Semiconductors **PL Mobile Audio** 411 E. Plumeria drive San Jose, CA 95134, USA E-mail: geert.knapen@nxp.com
James Goel	Qualcomm, Inc.
Andre Schevciw	Qualcomm, Inc.
Jin-Sheng Wang	Qualcomm, Inc.
Morten Christiansen	Synopsys

REVISION HISTORY

Revision	Date	Filename	Description
1.0	Mar. 18, 98	Frmts10.pdf	Release 1.0
2.0	May. 31, 06	Frmts20 final.pdf	Release 2.0
3.0	Sep. 22, 16	Frmts30.pdf	Release 3.0

Copyright © 1997-2016 USB Implementers Forum, Inc.
All rights reserved.

INTELLECTUAL PROPERTY DISCLAIMER

A LICENSE IS HEREBY GRANTED TO REPRODUCE THIS SPECIFICATION FOR INTERNAL USE ONLY. NO OTHER LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, IS GRANTED OR INTENDED HEREBY.

USB-IF AND THE AUTHORS OF THIS SPECIFICATION EXPRESSLY DISCLAIM ALL LIABILITY FOR INFRINGEMENT OF INTELLECTUAL PROPERTY RIGHTS RELATING TO IMPLEMENTATION OF INFORMATION IN THIS SPECIFICATION. USB-IF AND THE AUTHORS OF THIS SPECIFICATION ALSO DO NOT WARRANT OR REPRESENT THAT SUCH IMPLEMENTATION(S) WILL NOT INFRINGE THE INTELLECTUAL PROPERTY RIGHTS OF OTHERS.

THIS SPECIFICATION IS PROVIDED “AS IS” AND WITH NO WARRANTIES, EXPRESS OR IMPLIED, STATUTORY OR OTHERWISE. ALL WARRANTIES ARE EXPRESSLY DISCLAIMED. USB-IF, ITS MEMBERS AND THE AUTHORS OF THIS SPECIFICATION PROVIDE NO WARRANTY OF MERCHANTABILITY, NO WARRANTY OF NON-INFRINGEMENT, NO WARRANTY OF FITNESS FOR ANY PARTICULAR PURPOSE, AND NO WARRANTY ARISING OUT OF ANY PROPOSAL, SPECIFICATION, OR SAMPLE.

IN NO EVENT WILL USB-IF, MEMBERS OR THE AUTHORS BE LIABLE TO ANOTHER FOR THE COST OF PROCURING SUBSTITUTE GOODS OR SERVICES, LOST PROFITS, LOSS OF USE, LOSS OF DATA OR ANY INCIDENTAL, CONSEQUENTIAL, INDIRECT, OR SPECIAL DAMAGES, WHETHER UNDER CONTRACT, TORT, WARRANTY, OR OTHERWISE, ARISING IN ANY WAY OUT OF THE USE OF THIS SPECIFICATION, WHETHER OR NOT SUCH PARTY HAD ADVANCE NOTICE OF THE POSSIBILITY OF SUCH DAMAGES.

NOTE: VARIOUS USB-IF MEMBERS PARTICIPATED IN THE DRAFTING OF THIS SPECIFICATION. CERTAIN OF THESE MEMBERS MAY HAVE DECLINED TO ENTER INTO A SPECIFIC AGREEMENT LICENSING INTELLECTUAL PROPERTY RIGHTS THAT MAY BE INFRINGED IN THE IMPLEMENTATION OF THIS SPECIFICATION. PERSONS IMPLEMENT THIS SPECIFICATION AT THEIR OWN RISK.

Dolby™, AC-3™, Pro Logic™ and Dolby Surround™ are trademarks of Dolby Laboratories, Inc.
All other product names are trademarks, registered trademarks, or service marks of their respective owners.

Please send comments via electronic mail to audio-chair@usb.org

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1 INTRODUCTION

The intention of this document is to describe in detail all the Audio Data Formats that are supported by the Audio Device Class. This document is considered an integral part of *the Audio Device Class Specification*, although subsequent revisions of this document are independent of the revision evolution of the main *USB Audio Specification*. This is to easily accommodate the addition of new Audio Data Formats without impeding the core *USB Audio Specification*.

1.1 RELATED DOCUMENTS

- *Universal Serial Bus Specification*, Revision 2.0 (referred to in this document as the *USB Specification*). In particular, see Chapter 5, "USB Data Flow Model" and Chapter 9, "USB Device Framework."
- Universal Serial Bus Device Class Definition for Audio Devices (referred to in this document as USB Audio Device Class).
- Universal Serial Bus Device Class Definition for Terminal Types (referred to in this document as USB Audio Terminal Types).
- ANSI S1.11-1986 standard.
- MPEG-1 standard ISO/IEC 111172-3 1993. (available from http://www.iso.ch)
- MPEG-2 standard ISO/IEC 13818-3 Feb. 20, 1997. (available from http://www.iso.ch)
- Digital Audio Compression Standard (AC-3), ATSC A/52A Aug. 20, 2001. (available from http://www.atsc.org)
- Windows Media Audio (WMA) specification. (available from http://www.microsoft.com)
- ANSI/IEEE-754 floating-point standard.
- ISO/IEC 60958 International Standard: *Digital Audio Interface and Annexes.*
- ISO/IEC 61937 standard.
- ITU G.711 standard.
- ETSI Specification TS 102 114, "DTS Coherent Acoustics; Core and Extensions". (Available from http://webapp.etsi.org/action%5CPU/20020827/ts_102114v010101p.pdf)
- HDCP LINK TO BE ADDED

1.2 TERMS AND ABBREVIATIONS

This section defines terms used throughout this document. For additional terms that pertain to the Universal Serial Bus, see Chapter 2, "Terms and Abbreviations," in the *USB Specification*.

AC-3	Audio compression standard from Dolby Labs.
Audio Slot	A collection of audio subslots, each containing a PCM audio sample of a different physical audio channel, taken at the same moment in time.
Audio Stream	A concatenation of a potentially very large number of audio slots ordered according to ascending time.
Audio Subslot	Holds a single PCM audio sample.
DTS	Acronym for Digital Theater Systems.
DVD	Acronym for Digital Versatile Disc.
Encoded Audio Bit Stream	A concatenation of a potentially very large number of encoded audio frames, ordered according to ascending time.

Encoded Audio Frame	A sequence of bits that contains an encoded representation of audio samples from one or more physical audio channels taken over a fixed period of time.
MPEG	Acronym for Moving Pictures Expert Group.
PCM	Acronym for Pulse Coded Modulation.
Service Interval	A grouping of USB (micro)frames or Bus Intervals that are related.
Service Interval Packet	A packet that contains all the audio slots that are transferred over the bus during a Service Interval.
Transfer Delimiter	A unique token that indicates an interruption in an isochronous data packet stream. Can be either a zero-length data packet or the absence of an isochronous transfer in a certain USB frame.
WMA	Acronym for Windows Media Audio.

2 AUDIO DATA FORMATS

Audio Data formats can be divided in two main groups:

- Simple Audio Data Formats
- Extended Audio Data Formats

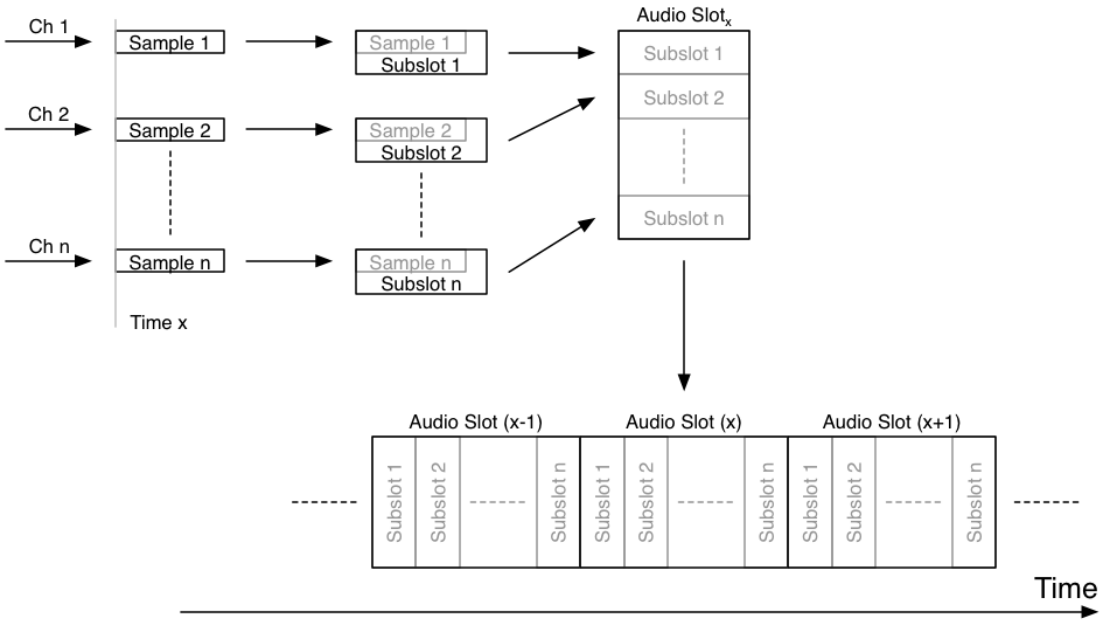
Simple Audio Data Formats can then be subdivided into three groups according to type.

The first group, Type I, deals with audio data streams that are transmitted over USB and are constructed on a sample-by-sample basis. Each audio sample is represented by a single independent symbol, contained in an audio subslot. Different compression schemes may be used to transform the audio samples into symbols.

Note: This is different from encoding. Compression is considered to take place on a per-audio-sample base. Each audio sample generates one symbol (e.g. A-law compression where a 16-bit audio sample is compressed into an 8-bit symbol).

If multiple physical audio channels are formatted into a single audio channel cluster, then samples at time *x* of subsequent channels are first contained into audio subslots. These audio subslots are then interleaved, according to the cluster channel ordering as described in the main *USB Audio Specification*, and then grouped into an audio slot. The audio samples, taken at time *x*+1, are interleaved in the same fashion to generate the next audio slot and so on. The notion of physical channels is explicitly preserved during transmission. A typical example of Type I formats is the standard PCM audio data. The following figure illustrates the concept.

Figure 2-1: Type I Audio Stream



The second group, Type III, contains Audio Data Formats that use encapsulation as described in the ISO/IEC 61937 standard before being sent over USB. One or more non-PCM encoded audio data streams are packed into "pseudo-stereo samples" and transmitted as if they were real stereo PCM audio samples. The sampling frequency of these pseudo samples (transport sampling frequency, as reported by the Clock Frequency Control of the associated Clock Source Entity) either matches the sampling frequency of the original non-encoded PCM audio data streams (native sampling frequency) or there is an integer ratio relationship between them. Therefore, clock recovery at the receiving end is relatively easy. The drawback is that unless multiple non-PCM encoded streams are packed into one pseudo stereo stream, more bandwidth than necessary is consumed.

The third group, Type IV, deals with audio streams that are not transmitted over USB. Instead, they interface with the Audio Function through an AudioStreaming interface that does not contain a USB isochronous IN or OUT endpoint. These streams typically connect via a digital interface like S/PDIF (or some other means of connectivity) and may require interaction from the Host before they enter or leave the Audio Function. A typical example is an external S/PDIF connector that can accept an AC-3 encoded audio stream. This stream is first processed by an AC-3 decoder before the (decoded) logical audio channels enter the Audio Function through the Input Terminal that represents this S/PDIF connection.

In addition to the Simple Audio Data Formats described above, Extended Audio Data Formats are defined. These are based on the Simple Audio Data Formats Type I and III definitions but they provide an optional packet header and for the Extended Audio Data Format Type I, an optional synchronous (i.e. sample accurate) control channel. Type IV Audio Data Formats do not have an Extended Audio Data Format definition.

The following sections explain the different Audio Data Formats and Format Types in more detail.

2.1 TRANSFER DELIMITER

Isochronous data streams are continuous in nature, although the actual number of bytes sent per packet may vary throughout the lifetime of the stream (for rate adaptation purposes for instance). To indicate a temporary stop in the isochronous data stream without closing the pipe (and thus relinquishing the USB bandwidth), an in-band Transfer Delimiter needs to be defined. This specification considers two situations to be a Transfer Delimiter. The first is a zero-length data packet and the second is the absence of an isochronous transfer in a USB (micro)frame that would normally have an isochronous transfer. Both situations are considered equivalent and the Audio Function is expected to behave the same. However, the second type consumes less isochronous USB bandwidth (i.e. zero bandwidth). In both cases, this specification considers a Transfer Delimiter to be an entity that can be sent over the USB.

2.2 SERVICE INTERVAL AND SERVICE INTERVAL PACKET DEFINITIONS

Note: The *USB Audio 2.0 Specification* used the terms Virtual Frame and Virtual Frame Packet to describe the same concepts that were called Service Interval and Service Interval Packet in the *USB 3.0 and higher Specifications*. In this specification, the terminology has been consolidated to use the more general terms of Service Interval and Service Interval Packet instead of the USB Audio 2.0-specific terminology of Virtual Frame and Virtual Frame Packet. Also, the term Bus Interval is used instead of USB (micro)frame, wherever applicable.

In the following paragraphs, the packetizing process for audio is described in terms of Service Interval and Service Interval Packets. This provides a consistent model of 'one Service Interval Packet (SIP) per Service Interval (SI)', irrespective of the actual transactions on the USB and the version of USB used.

A Service Interval is defined as:

$$Service\ Interval = Bus\ Interval * 2^{(bInterval-1)}$$

where Bus Interval has a value of 1 ms for full-speed isochronous endpoints and 125 μs for high-speed, SuperSpeed, and Enhanced SuperSpeed isochronous endpoints and where bInterval is the value specified in the bInterval field of the standard endpoint descriptor.

A Service Interval Packet is defined as the amount of isochronous data that is transported during an entire Service Interval. For high-speed high-bandwidth endpoints, the Service Interval Packet is the concatenation of the two or three physical packets that are transferred over the bus in a Bus Interval.

Note: The *USB Specification* already considers the 2 or 3 transactions of a high-speed high-bandwidth transfer to be part of a single packet.

2.3 SIMPLE AUDIO DATA FORMATS

2.3.1 TYPE I FORMATS

The following sections describe the Audio Data Formats that belong to Type I. A number of terms and their definition are presented.

2.3.1.1 USB PACKETS

Audio data streams that are inherently continuous shall be packetized when sent over the USB. The quality of the packetizing algorithm directly influences the amount of effort needed to reconstruct a reliable sample clock at the receiving side.

Furthermore, the chosen size of the Service Interval has a direct impact on the amount of buffer memory needed on both sides of the pipe and also on the incurred latency over the pipe. Shorter Service Intervals minimize buffer requirements and therefore also latency at the potential expense of higher power consumption. Indeed, longer Service Intervals potentially allow the bus (and parts of the sender and receiver's hardware) to enter lower power states for longer periods of time, thus conserving more power.

This specification defines two possible modes to transport audio data streams over the USB:

- Continuous Mode
- Burst Mode

Continuous Mode occurs when the Service Interval is smaller or equal to one USB Frame time of 1 ms. This mode minimizes buffer requirements and latency at the potential expense of higher power consumption. Note however that choosing a Service Interval that is smaller than one USB Frame time may result in excessive system level interrupts.

Burst Mode occurs when the Service Interval is larger than one USB Frame time. This mode provides for opportunities to save more power by allowing various system components to enter low power states for extended periods of time at the expense of larger buffer sizes and increased latency.

Devices may choose to expose multiple Alternate Settings of their AudioStreaming interface(s) with different Service Interval settings for each Alternate Setting, thus allowing the Host to choose a setting that best fits the desired use case. However, all devices shall expose at least one Alternate Setting (besides the zero bandwidth Alternate setting 0) that supports Continuous Mode (Service Interval <= 1 ms).

2.3.1.1.1 SERVICE INTERVAL PACKET SIZE CALCULATION

The goal shall be to keep the instantaneous number of audio slots per SI, n_i as close as possible to the average number of audio slots per SI, n_{av}. The average n_{av} shall be calculated as follows:

$$n_{av} = \frac{T_{SI}}{\Delta t}$$

where T_{SI} is the duration of an SI and Δt is the sample time $(1/F_s)$. In most cases, n_{av} will be a number with a fractional part.

If the sampling rate is a constant, the allowable variation on n_i is limited to one audio slot, that is, $\Delta n_i = 1$. This implies that all SIPs shall either contain $INT(n_{av})$ audio slots (small SIP) or $INT(n_{av}) + 1$ (large SIP) audio slots. For all i:

$$n_i = INT(n_{av}) | INT(n_{av}) + 1$$

Note: In the case where $n_{av} = INT(n_{av})$, n_i may vary between $INT(n_{av}) - 1$ (small SIP), $INT(n_{av})$ (medium SIP) and $INT(n_{av}) + 1$ (large SIP).

Furthermore, a large SIP shall be generated as soon as it becomes available. Typically, a source will generate a number of small SIPs as long as the accumulated fractional part of n_{av} remains < 1. Once the accumulated fractional part of n_{av} becomes ≥ 1, the source shall send a large SIP and decrement the accumulator by 1.

Due to possible different notions of time in the source and the sink (they might each have their own independent sampling clock), the (small SIP)/(large SIP) pattern generated by the source may be different from what the sink expects. Therefore, the sink shall be capable to accept a large SIP at all times.

Example:

Assume F_S = 44,100 Hz and T_{VF} = 1ms. Then n_{av} = 44.1 audio slots. Since the source can only send an integer number of audio slots per SI, it will send small SIPs of 44 audio slots. Each SI, it therefore sends '0.1 slot' too few and it will accumulate this fractional part in an accumulator. After having sent 9 small SIPs of 44 audio slots, at the tenth SI it will have exactly one audio slot in excess and therefore can send a large SIP containing 45 audio slots. Decrementing the accumulator by 1 brings it back to 0 and the process can start all over again. The source will thus produce a repetitive pattern of 9 small SIPs of 44 audio slots followed by 1 large SIP of 45 audio slots. The following table illustrates the process:

Table 2-1: Packetization

#SI	n_{av}	n_i	Fraction	Accumulator
n	44.1	44	0.1	0.1
n+1	44.1	44	0.1	0.2
n+2	44.1	44	0.1	0.3
n+3	44.1	44	0.1	0.4
n+4	44.1	44	0.1	0.5
n+5	44.1	44	0.1	0.6
n+6	44.1	44	0.1	0.7
n+7	44.1	44	0.1	0.8
n+8	44.1	44	0.1	0.9
n+9	44.1	**45**	0.1	1.0 -> 0
n+10	44.1	44	0.1	0.1
n+11	44.1	44	0.1	0.2
…	…	…	…	…

2.3.1.2 PITCH CONTROL

If the sampling rate can be varied (to implement pitch control), the allowable variation on n_i is limited to one audio slot per SI. For all i:

$$n_{i+1} = n_i | n_i \pm 1$$

Pitch control is restricted to adaptive endpoints only. AudioStreaming interfaces that support pitch control on their isochronous endpoint are required to report this in the class-specific endpoint descriptor. In addition, a Set/Get Pitch Control request is required to enable or disable the pitch control functionality.

2.3.1.3 AUDIO SUBSLOT

The basic structure used to represent audio data is the audio subslot. An audio subslot holds a single audio sample. An audio subslot always contains an integer number of bytes.

This specification limits the possible audio subslot sizes (**bSubslotSize**) to 1, 2, 3, 4, or 8 bytes per audio subslot. An audio sample is represented using a number of bits (**bBitResolution**) less than or equal to the total number of bits available in the audio subslot, i.e. **bBitResolution** ≤ **bSubslotSize***8.

AudioStreaming endpoints shall be constructed in such a way that a valid transfer can take place as long as the reported audio subslot size (**bSubslotSize**) is respected during transmission. If the reported bits per sample (**bBitResolution**) do not correspond with the number of significant bits actually used during transfer, the device will either discard trailing significant bits ([actual_bits_per_sample] > **bBitResolution**) or interpret trailing zeroes as significant bits ([actual_bits_per_sample] < **bBitResolution**).

2.3.1.4 AUDIO SLOT

An audio slot consists of a collection of audio subslots, each containing an audio sample of a different physical audio channel, taken at the same moment in time. The number of audio subslots in an audio slot equals the number of logical audio channels in the audio channel cluster. The ordering of the audio subslots in the audio slot obeys the rules set forth in the *USB Audio Specification*. All audio subslots shall have the same audio subslot size.

2.3.1.5 AUDIO STREAMS

An audio stream is a concatenation of a potentially very large number of audio slots, ordered according to ascending time. Streams are packetized when transported over USB whereby SIPs can only contain an integer number of audio slots. Each packet always starts with the same channel, and the channel order is respected throughout the entire transmission. If, for any reason, there are no audio slots available to construct a SIP, a Transfer Delimiter shall be sent instead.

2.3.1.6 TYPE I SUPPORTED FORMATS

The following paragraphs list all currently supported Type I Audio Data Formats. The bit allocations in the **bmFormats** field of the class-specific AS interface descriptor for the different Type I Audio Data Formats can be found in Appendix A.1, “Audio Data Formats Bit Allocations.” Note that an Alternate Setting of an AudioStreaming interface can only support one Type I Audio Data Format. Consequently, only one Type I Audio Data Format bit shall be set in the **bmFormats** field of the class-specific AS interface descriptor. It is allowed to support one or more Type III Audio Data Formats in the same AudioStreaming interface if the interface is able to independently distinguish between the Type I Audio Data Format and any other Type III Audio Data Format.

2.3.1.6.1 PCM FORMAT

The PCM (Pulse Coded Modulation) format is the most commonly used audio format to represent audio data streams. The audio data is not compressed and uses a signed two’s-complement fixed point format. It is left-justified (the sign bit is the Msb) and data is padded with trailing zeroes to fill the remaining unused bits of the subslot. The binary point is located to the right of the sign bit so that all values lie within the range [-1, +1).

2.3.1.6.2 PCM8 FORMAT

The PCM8 format is introduced to be compatible with the legacy 8-bit wave format. Audio data is uncompressed and uses 8 bits per sample (**bBitResolution** = 8). In this case, data is unsigned fixed-point, left-justified in the audio subslot, Msb first. The range is [0,255].

2.3.1.6.3 IEEE_FLOAT FORMAT

The IEEE_FLOAT format is based on the ANSI/IEEE-754 floating-point standard. Audio data is represented using the basic single-precision format. The basic single-precision number is 32 bits wide and has an 8-bit exponent and a 24-bit mantissa. Both mantissa and exponent are signed numbers, but neither is represented in two's-complement format. The mantissa is stored in sign magnitude format and the exponent in biased form (also called excess-n form). In biased form, there is a positive integer (called the bias) which is subtracted from the stored number to

get the actual number. For example, in an eight-bit exponent, the bias is 127. To represent 0, the number 127 is stored. To represent -100, 27 is stored. An exponent of all zeroes and an exponent of all ones are both reserved for special cases, so in an eight-bit field, exponents of -126 to +127 are possible. In the basic floating-point format, the mantissa is assumed to be normalized so that the most significant bit is always one, and therefore is not stored. Only the fractional part is stored. Denormalized (exponent = 0) values are considered to be zero.

The 32-bit IEEE-754 floating-point word is broken into three fields. The most significant bit stores the sign of the mantissa, the next group of 8 bits stores the exponent in biased form, and the remaining 23 bits store the magnitude of the fractional portion of the mantissa. For further information, refer to the ANSI/IEEE-754 standard.

The data is conveyed over USB using 32 bits per sample (**bBitResolution** = 32; **bSubslotSize** = 4).

2.3.1.6.4 ALAW FORMAT AND μLAW FORMAT

Starting from 12- or 16-bits linear PCM samples, simple compression down to 8-bits per sample (one byte per sample) can be achieved by using logarithmic companding. The compressed audio data uses 8 bits per sample (**bBitsPerSample** = 8). Data is signed fixed point, left-justified in the subslot, Msb first. The compressed range is [-128,128]. The difference between Alaw and μLaw compression lies in the formulae used to achieve the compression. Refer to the ITU G.711 standard for further details.

2.3.1.6.5 DSD FORMAT

The Direct-Stream Digital (DSD) format uses pulse-density modulation encoding—a technology primarily used to store audio signals on SACD (Super Audio CD) digital storage media.

Audio data is stored as single-bit delta-sigma modulated digital audio; i.e. a sequence of single-bit values at a sampling rate of 2.8224 MHz (64 times the CD audio sampling rate of 44.1 kHz) for basic sampling rate DSD64, 5.6448 MHz for DSD128 (2X-rate DSD), 11.2896 MHz for DSD256 (4X-rate DSD), 22.5792 MHz for DSD 512 (8X-rate DSD), and 45.1584 MHz for DSD1024 (16X-rate DSD). 48 kHz-based DSD streams are also in existence. In that case, the bitstream sampling rates are 3.072 MHz, 6.144 MHz, 12.288 MHz, 24.576 MHz, and 49.152 MHz respectively.

No matter what sampling rate the DSD stream uses, the audio subslot size is fixed to 64 bits (**bBitResolution** = 64; **bSubslotSize** = 8) so that, at the transport layer, the DSD stream always looks like 64-bit PCM data. Therefore, the transport sampling rate of the DSD stream, packetized as 64-bit PCM samples, is always 1/64 of the DSD sampling rate and the Clock Source Entity, connected to the Terminal, representing the DSD AudioStreaming interface shall always advertise this transport sampling rate:

- 44.1 kHz, 88.2 kHz, 176.4 kHz, 352.8 kHz, or 705.6 kHz for 44.1 kHz based DSD streams
- 48 kHz, 96 kHz, 192 kHz, 384 kHz, or 768 kHz for 48 kHz based DSD streams

2.3.1.6.6 TYPE I RAW DATA

This audio format is included to allow transport of data (audio or other) over a USB AudioStreaming interface in the form of PCM-like audio slots when the actual format or even the meaning of the transported data is unknown. The USB pipe simply acts as a pass-through. As a consequence, such data can never be interpreted inside the Audio Function and can only be routed from an Input Terminal to one or more Output Terminals. From a USB standpoint, the data is packed as if it were Type I formatted audio data, but the data is never to be interpreted as being audio data.

2.3.2 TYPE III FORMATS

These formats are based upon the IEC61937 standard. The IEC61937 standard describes a method to transfer non-PCM encoded audio bit streams over an IEC60958 digital audio interface, together with the transfer of the accompanying “Channel Status” and “User Data.”

The IEC60958 standard specifies a widely used method of interconnecting digital audio equipment with two-channel linear PCM audio. The IEC61937 standard describes a way in which the IEC60958 interface shall be used to convey non-PCM encoded audio bit streams for consumer applications.

The same basic techniques used in IEC61937 are reused here to convey non-PCM encoded audio bit streams over a Type III formatted audio stream. From a USB transfer standpoint, the data streaming over the interface looks exactly like two-channel 16 bit PCM audio data.

2.3.2.1 TYPE III SUPPORTED FORMATS

Refer to the ISO/IEC 60958 and ISO/IEC 61937 (several parts) specifications for detailed format information. The bit allocations in the **bmFormats** field of the class-specific AS interface descriptor for the different Type III Audio Data Formats can be found in Appendix A.1, "Audio Data Formats Bit Allocations."

Note that an Alternate Setting of an AudioStreaming interface can support multiple Type III Audio Data Formats. Consequently, multiple bits can be simultaneously set in the **bmFormats** field of the class-specific AS interface descriptor. It is then up to the decoder to inspect the encoded data stream to determine exactly which format is currently being used over the pipe. Alternatively, a device may expose multiple Alternate Settings on an AudioStreaming interface, each advertising support for only a single Type III Audio Data Format. The Host can then select an appropriate Alternate Setting that corresponds to the format currently in use.

The following is a list of formats that is covered by the above specifications.

- PCM_IEC60958
- AC-3
- MPEG-1_Layer1
- MPEG-1_Layer2/3 or MPEG-2_NOEXT
- MPEG-2_EXT
- MPEG-2_AAC_ADTS
- MPEG-2_Layer1_LS
- MPEG-2_Layer2/3_LS
- DTS-I
- DTS-II
- DTS-III
- ATRAC
- ATRAC2/3
- WMA
- E-AC-3
- MAT
- DTS-IV
- MPEG-4_HE_AAC
- MPEG-4_HE_AAC_V2
- MPEG-4_AAC_LC
- DRA
- MPEG-4_HE_AAC_SURROUND
- MPEG-4_AAC_LC_SURROUND
- MPEG-H_3D_AUDIO
- AC4
- MPEG-4_AAC_ELD

2.3.3 TYPE IV FORMATS

Type IV formats can only be used on external connections to the Audio Function that do not use a USB pipe for their data transport but that do need an AudioStreaming interface to advertise their presence. A typical example of such a connection is an S/PDIF connector that is capable of handling both PCM stereo audio data streams (IEC60958) in one Alternate Setting and encoded data streams (IEC61937) in another Alternate Setting of the interface. Note however that the external connection could also be vendor specific (like a parallel data interface).

2.3.3.1 TYPE IV SUPPORTED FORMATS

This specification supports all Audio Data Formats on an external connection that are defined on a USB pipe (Type I and Type III). See Section 2.3.1.6, “Type I Supported Formats”, and Section 2.3.2.1, “Type III Supported Formats”.

The bit allocations in the **bmFormats** field of the class-specific AS interface descriptor for the different Type IV Audio Data Formats can be found in Appendix A.1, “Audio Data Formats Bit Allocations.”

Note that an Alternate Setting of an AudioStreaming interface can support multiple Type IV Audio Data Formats. Consequently, multiple bits can be simultaneously set in the **bmFormats** field of the class-specific AS interface descriptor. It is then up to the decoder to inspect the encoded data stream to determine exactly what format is currently being used over the connection.

2.4 EXTENDED AUDIO DATA FORMATS

In addition to the Simple Audio Data Formats described above, Extended Audio Data Formats Type I and III are defined. These are based on the Simple Audio Data Formats Type I and III definitions but they provide an optional Header and for the Extended Audio Data Format Type I, an optional synchronous (i.e. sample accurate) Control Stream.

Each SIP shall start with a SIPDescriptor, followed by the following optional components:

- Header
- Audio data, formatted according the Simple Audio Data Formats Type I or III
- Synchronous vendor-specific Control Stream (only for Extended Audio Format Type I)

These three components can be optionally present on a per-SIP basis.

The SIPDescriptor is exactly 4 bytes long and has the following layout:

The **bmFlags** field indicates which of the components are present in the SIP as follows:

- Bit D0 indicates whether a Header is present in the SIP (D0=0b1) or not (D0=0b0). When the Header is not present, the **wHeaderLength** field shall be set to zero (0x0000).
- Bit D1 indicates whether an AudioSlot is present in all of the Extended AudioSlots of the SIP (D1=0b1) or not (D1=0b0).
- Bit D2 indicates whether a Control Stream is present. In other words, it indicates whether a Control Word is present in all of the Extended AudioSlots of the Extended Type I SIP (D2=0b1) or not (D2=0b0). This bit shall be set to zero (D2=0b0) for Extended Type III formats.
- Bits D15..3 are reserved.

The **wHeaderLength** field indicates the total length of the Header in bytes. This includes all the SubHeaders that together make up the total Header.

The presence of the SIPDescriptor allows for maximum flexibility. For example, it is possible to create an Audio Stream consisting of a Control Stream only, without Header or audio data. It is also possible to create an Audio Stream where Headers are only present once in a while.

If present, the Header immediately follows the SIPDescriptor. There is only one Header allowed per SIP. The Header can consist of one or more SubHeaders, each having a different purpose. Each SubHeader shall start with a

wSubHeaderID field, uniquely identifying the purpose of the SubHeader. The length of each SubHeader and the semantics of the remaining fields in the SubHeader are defined by this specification. The structure and composition of the Header and the order in which the SubHeaders appear in the Header can change from SIP to SIP.

Table 2-1: SIPDescriptor Layout

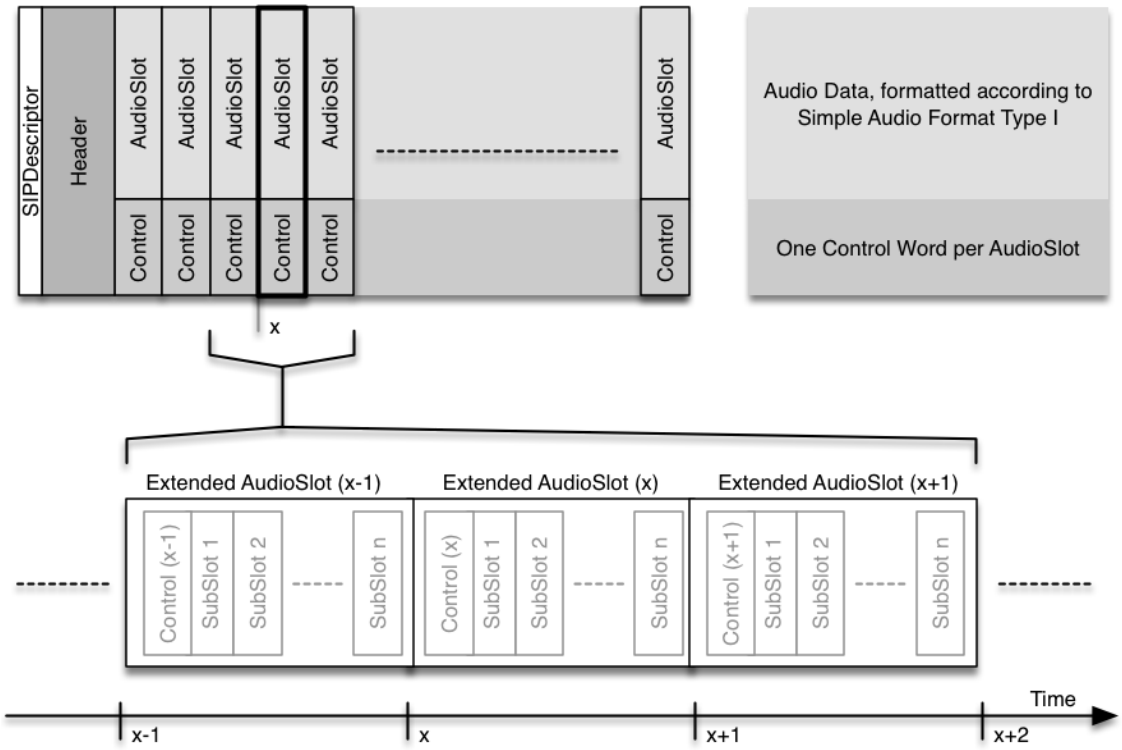
Offset	Field	Size	Value	Description
0	bmFlags	2	Bitmap	D0: Header present. D1: AudioSlot present. D2: Control Stream present. D15..3: Reserved.
2	wHeaderLength	2	Number	Total length of the Header, in bytes.

2.4.1 EXTENDED TYPE I FORMATS

The following figure illustrates the possible layout of a SIP. The greyed parts are optional.

Note: In order to have a meaningful stream, at least one of the optional components shall be present.

Figure 2-2: Extended Type I Format



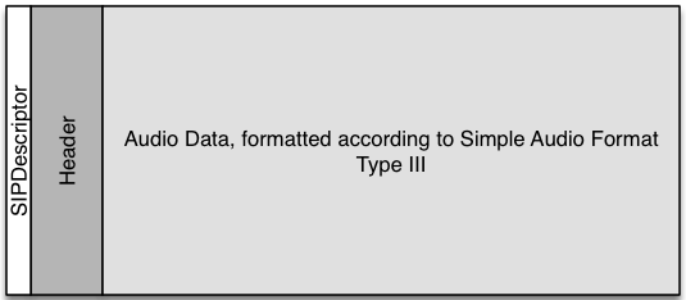
An Extended AudioSlot is the concatenation of a Control Word, followed by the Type I AudioSlot. The Control Stream therefore consists of a sequence of Control Words, where each Control Word is synchronous to its associated AudioSlot. There are as many Control Words per SIP as there are AudioSlots in the SIP. The byte size of the Control Words is independent of the AudioSubSlot size and is the same for each AudioSlot.

2.4.2 EXTENDED TYPE III FORMATS

The following figure illustrates the possible layout of a SIP. The greyed parts are optional.

Note: In order to have a meaningful stream, at least one of the optional components should be present.

Figure 2-3: Extended Type III Format



The Header is optionally followed by the actual encoded audio frame data.

2.5 CLASS-SPECIFIC AS INTERFACE DESCRIPTOR

The first eight fields of this descriptor are described in the USB Audio Device Class Definition document.

The **bSize** field indicates the length in bytes of the **bmFormats** field. The **bmFormats** field consists of an array of bits where each bit indicates a particular Audio Data Format is supported (Dx = 1) or not (Dx = 0).

It is allowed to support both Type I and Type III formats in the same Alternate Setting of an AudioStreaming interface as long as the interface is able to unambiguously determine which exact format is currently in use by inspecting the data in the audio stream. Type IV formats can never be mixed with either Type I or Type III formats in the same AudioStreaming interface. Type IV formats are distinguished from Type I/Type III formats by the absence of a USB endpoint in the AudioStreaming interface.

A complete list of currently supported Audio Data Formats, their bit allocation in the **bmFormats** field and their usage can be found in Appendix A.1, “Audio Data Formats Bit Allocations.”

For Simple and Extended Type I formats, the **bSubslotSize** field can be set to 1, 2, 4, or 8, and the **bBitResolution** field indicates how many bits of the total number of available bits in the audio subslot are effectively used to convey audio information.

For Simple and Extended Type III formats, the **bSubslotSize** field shall be set to 2 and the **bBitResolution** field shall be set to 16.

For Type IV formats, the **bSubslotSize** and **bBitResolution** fields are not used and shall be set to zero.

For Simple Type I and Type III formats, the **bmAuxProtocols** and **bControlSize** fields are not used and shall be set to zero.

For Extended Type I and Extended Type III formats, this specification defines a number of Auxiliary Protocols (See Section 3, “Auxiliary Protocols”). The **bmAuxProtocols** field contains a bitmap identifying which Auxiliary Protocols this AudioStreaming interface’s Alternate Setting requires. For each Auxiliary Protocol, the AudioStreaming interface may offer up to two Alternate Settings, one in which the Auxiliary Protocol is required and the other in which it is not. For example, an AudioStreaming Interface may offer two Alternate Settings, one indicating the required use of HDCP and the other indicating that it operates without HDCP.

If the stream does not contain actual audio data, the **bNrChannels**, **bmChannelConfig** and **iChannelNames** in the class-specific AS Interface descriptor (See the *USB Audio Device Class* document) shall all be set to zero.

For Extended Type I formats, the **bControlSize** field indicates the size in bytes of each Control Channel Word in the stream. It shall be set to zero for all other formats.

Table 2-2: Class-Specific AS Interface Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	Number	Size of this descriptor in bytes: 23.
1	bDescriptorType	1	Constant	CS_INTERFACE descriptor type.
2	bDescriptorSubtype	1	Constant	AS_GENERAL descriptor subtype.
3	bTerminalLink	1	Number	The Terminal ID of the Terminal to which this interface is connected.
4	bmControls	4	Bitmap	D1..0: Active Alternate Setting Control. D3..2: Valid Alternate Settings Control. D5..4: Audio Data Format Control. D31..6: Reserved.
8	wClusterDescrID	2	Number	ID of the cluster descriptor of the AS Interface.
10	bmFormats	8	Bitmap	The Audio Data Format(s) that can be used to communicate with this interface.
18	bSubslotSize	1	Number	The number of bytes occupied by one audio subslot.
19	bBitResolution	1	Number	The number of effectively used bits from the available bits in an audio subslot.
20	bmAuxProtocols	2	Bitmap	Bitmap, indicating which Auxiliary Protocols are required.
22	bControlSize	1	Number	Size of the Control Channel Words, in bytes.

3 AUXILIARY PROTOCOLS

3.1 HDCP PROTOCOL

The HDCP protocol uses the HDCP SubHeader to convey the periodic Link Synchronization information, required by [HDCP2.2]. [HDCP2.2] requires that a 32-bit *streamCtr* value and a 64-bit *inputCtr* value be sent periodically from the content transmitter to the content receiver. The Header construct in the Extended Format Types shall be used to convey that information periodically as follows.

The HDCP SubHeader has the following layout:

Table 3-1: HDCP SubHeader Layout

Offset	Field	Size	Value	Description
0	bLength	1	Number	Size of this SubHeader, in bytes. 16.
1	bSubHeaderID	1	Number	HDCP_ENCRYPTION
2	bOffset	1	Number	Offset to first full 16-byte encrypted data block in the SIP.
3	bReserved	1	Number	Reserved for future extensions.
4	dStreamCtr	4	Number	The *streamCtr* value as assigned by the HDCP transmitter.
8	qInputCtr	8	Number	The *inputCtr* value associated with the first full 16-byte encrypted data block in the SIP.

The **bLength** field contains the size of this SubHeader, expressed in bytes.

The **bSubHeaderID** field contains the HDCP_ENCRYPTION constant to uniquely identify this SubHeader as an HDCP SubHeader.

The **bOffset** field contains an offset value ranging from 0 to 15, indicating at which byte position, measured from the start of the audio data in the SIP, the first full 16-byte encrypted data block (HDCP Cipherblock) starts. The value in the **qInputCtr** field shall pertain to that same 16-byte encrypted data block.

Note: Since an HDCP Cipher Block is always 16 bytes or 128 bits long, the start of a full Cipher Block is guaranteed to occur within the first 16 bytes of the actual audio data payload. Also, when a Control Stream is present, the **bOffset** field value only pertains to the actual audio data bytes (the Control Stream is not encrypted) and the Control Stream should be separated from the actual audio data bytes before the offset is applied.

The **bReserved** field is reserved and shall be set to zero.

The **dStreamCtr** field contains the *streamCtr* value associated with this particular stream as assigned by the HDCP transmitter.

The **qInputCtr** field contains the *inputCtr* value associated with the first full 16-byte encrypted audio data block in the SIP.

The HDCP SubHeader shall be present at least every HDCP_PACKET_HEADER_TIME but may occur more frequently. The presence of the HDCP SubHeader indicates to the HDCP receiver that the content is HDCP-encrypted.

4 ADDING NEW AUDIO DATA FORMATS

Adding new Audio Data Formats to this specification is achieved by proposing a fully documented Audio Data Format to the Audio Device Class Working Group. Upon acceptance, they will register the new Audio Data Format (attribute a unique bit position in the **bmFormats** field of the class-specific AS interface descriptor) and update this document accordingly. This process will also guarantee that new releases of generic USB audio drivers will support the newly registered Audio Data Formats.

It is always possible to use vendor-specific definitions if the above procedure is considered unsatisfactory.

5 ADDING NEW SIDE BAND PROTOCOLS

Adding new Side Band Protocols to this specification is achieved by proposing a fully documented Side Band Protocol to the Audio Device Class Working Group. Upon acceptance, they will register the new Side Band Protocol (attribute a unique SideBandProtocol constant) and update this document accordingly. This process will also guarantee that new releases of generic USB audio drivers will support the newly registered Side Band Protocols.

It is always possible to use vendor-specific definitions if the above procedure is considered unsatisfactory.

Appendix A. Additional Audio Device Class Codes

A.1 AUDIO DATA FORMATS BIT ALLOCATIONS

Table A-2: Audio Data Formats Bit Allocations in the bmFormats Field and Usage

Name	bmFormats	Usage		
		Type I	Type III	Type IV
PCM	D0	X		X
PCM8	D1	X		X
IEEE_FLOAT	D2	X		X
ALAW	D3	X		X
MULAW	D4	X		X
DSD	D5	X		X
RAW_DATA	D6	X		X
PCM_IEC60958	D7		X	X
AC-3	D8		X	X
MPEG-1_Layer1	D9		X	X
MPEG-1_Layer2/3 or MPEG-2_NOEXT	D10		X	X
MPEG-2_EXT	D11		X	X
MPEG-2_AAC_ADTS	D12		X	X
MPEG-2_Layer1_LS	D13		X	X
MPEG-2_Layer2/3_LS	D14		X	X
DTS-I	D15		X	X
DTS-II	D16		X	X
DTS-III	D17		X	X
ATRAC	D18		X	X
ATRAC2/3	D19		X	X
WMA	D20		X	X
E-AC-3	D21		X	X
MAT	D22		X	X
DTS-IV	D23		X	X
MPEG-4_HE_AAC	D24		X	X
MPEG-4_HE_AAC_V2	D25		X	X
MPEG-4_AAC_LC	D26		X	X
DRA	D27		X	X
MPEG-4_HE_AAC_SURROUND	D28		X	X
MPEG-4_AAC_LC_SURROUND	D29		X	X

Name	bmFormats	Usage		
		Type I	Type III	Type IV
MPEG-H_3D_AUDIO	D30		X	X
AC4	D31		X	X
MPEG-4_AAC_ELD	D32		X	X

A.2 SUBHEADER CODES

Table A-3: SubHeader Codes

SubHeader Code	Value
HEADER_UNDEFINED	0x00
HDCP_ENCRYPTION	0x01
Reserved	0x02..0xFF

A.3 AUDIO FORMAT GENERAL CONSTANTS

Table A-4: General Constants

Constant Identifier	Value
HDCP_PACKET_HEADER_TIME	512 (ms)