Universal Serial Bus
Device Class Definition
for
Audio Data Formats

Release 2.0

May 31, 2006

Scope of This Release

This document is the Release 2.0 of this device class definition.

Contributors

Geert Knapen (Editor)	Philips Applied Technologies AppTech-USA 1101 McKay Drive M/S 16 San Jose, CA 95131 USA Phone: +1 (408) 474-8774 E-mail: geert.knapen@philips.com
Mike Kent	Roland Corporation
Kaoru Ishimine	Roland Corporation
Shoichi Kojima	Roland Corporation
Robert Gilsdorf	Creative Labs
Daniel (D.J.) Sisolak	Microsoft Corporation
Jack Unverferth	Microsoft Corporation
Niel Warren	Apple Computer, Inc.
Len Layton	C-Media Electronics
Mark Cookson	M-Audio

Revision History

Revision	Date	Filename	Author	Description
1.7	Mar 18, 98	Frmts17.doc	USB-IF DWG	Original Frmts.doc document opened for review.
1.7a	Oct. 24, 02	Frmts17a.doc	Geert Knapen	Identified areas for change.
1.7b	Dec 06, 02	Frmts17b.doc	DJ Sisolak	Updated for USB 2.0 Core Specification
1.7c	Dec 10, 02	Frmts17c.doc	DJ Sisolak	Make comments on the edits and accepted a number of changes.
1.7d	Feb. 05, 03	Frmts17d.doc	Geert Knapen	Reviewed and accepted additional changes.
1.7e	Feb. 07, 03	Frmts17e.doc	Geert Knapen	Completed cluster descriptors in Format descriptors. Added language for the sliding averaging window.
1.7e1	Feb. 19, 03	Frmts17e1.doc	Geert Knapen	Actually added language for USB packetization.
1.7f	Mar. 26, 03	Frmts17f.doc	Geert Knapen	Accepted all changes
1.7g	Apr. 07, 03	Frmts17g.doc	Geert Knapen	Major overhaul. Halfway through the RANGE implementation
1.7h	Jun. 03, 03	Frmts17h.doc	Geert Knapen	Accepted all the changes so far.
1.7i	Jun. 03, 03	Frmts17i.doc	Geert Knapen	Edited requests to reflect the RANGE attribute

Revision	Date	Filename	Author	Description
1.7j	Jul..11, 03	Frmts1ji.doc	Geert Knapen	Accepted all the changes, fixed a duplicate definition for D6
1.7k	Sep. 08, 03	Frmts17k.doc	Geert Knapen	Added RAW_DATA format
1.7l	Sep. 10, 03	Frmts17l.doc	Geert Knapen	Accepted all the changes
1.7m	Oct. 14, 03	Frmts17m.doc	Geert Knapen	Added CN to all requests. Added some Controls.
1.7n	Nov. 05, 03	Frmts17n.doc	Geert Knapen	Accepted all the changes.
1.7o	Nov. 17, 03	Frmts17o.doc	Geert Knapen	Removed all references to sampling frequencies in the format-specific descriptors.
1.7p	Dec. 01, 03	Frmts17p.doc	Geert Knapen	Accepted all the changes
1.7q	Dec. 12, 03	Frmts17q.doc	Geert Knapen	Introduced extended Format Types
1.7r	Feb. 04, 04	Frmts17r.doc	Geert Knapen	Accepted all changes
1.7s	Apr. 13, 04	Frmts17s.doc	Geert Knapen	Added new Type III codes. Added Hi-Res Timestamp Sideband Protocol. Added Type IV Format. Moved decoder information to Audio document. Removed the concept of Format-specific descriptors and replaced them with Decoder descriptors
1.7t	Apr. 28, 04	Frmts17t.doc	Geert Knapen	Added more info on the different audio data format types.
1.8	May 26, 04	Frmts18.doc	Geert Knapen	Accepted all changes and promoted to 1.8 level.
1.8a	Aug. 10, 05	Frmts18a.doc	Geert Knapen	Minor editorial changes
1.8b	Aug. 16, 05	Frmts18b.doc	Geert Knapen	Accepted editorial changes, based on F2F meeting review
1.8c	Aug. 16, 05	Frmts18c.doc	Geert Knapen	Added DTS support
1.8d	Sep. 02, 05	Frmts18d.doc	Geert Knapen	Accepted all the changes.
1.9RC1	Sep. 02, 05	Frmts19RC1.doc	Geert Knapen	Republished unchanged as 1.9RC1
1.9RC2	Oct. 05, 05	Frmts19RC2.doc	Geert Knapen	Removed comment on the Microsoft link. Accepted the change.
1.9	Oct. 07, 05	Frmts19.doc	Geert Knapen	Promoted to 1.9 without change.
2.0RC1	May 19, 06	Frmts20RC1.doc	Geert Knapen	Promoted to 2.0RC1 without change.

Revision	Date	Filename	Author	Description
2.0	May 31, 06	Frmts20.doc	Geert Knapen	Added new Intellectual Property Disclaimer. Final version.

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.

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)

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.
Virtual Frame	A grouping of USB (micro)frames that are related.
Virtual Frame Packet	**A** packet that contains all the audio slots that are transferred over the bus during a virtual frame.
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 four 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.

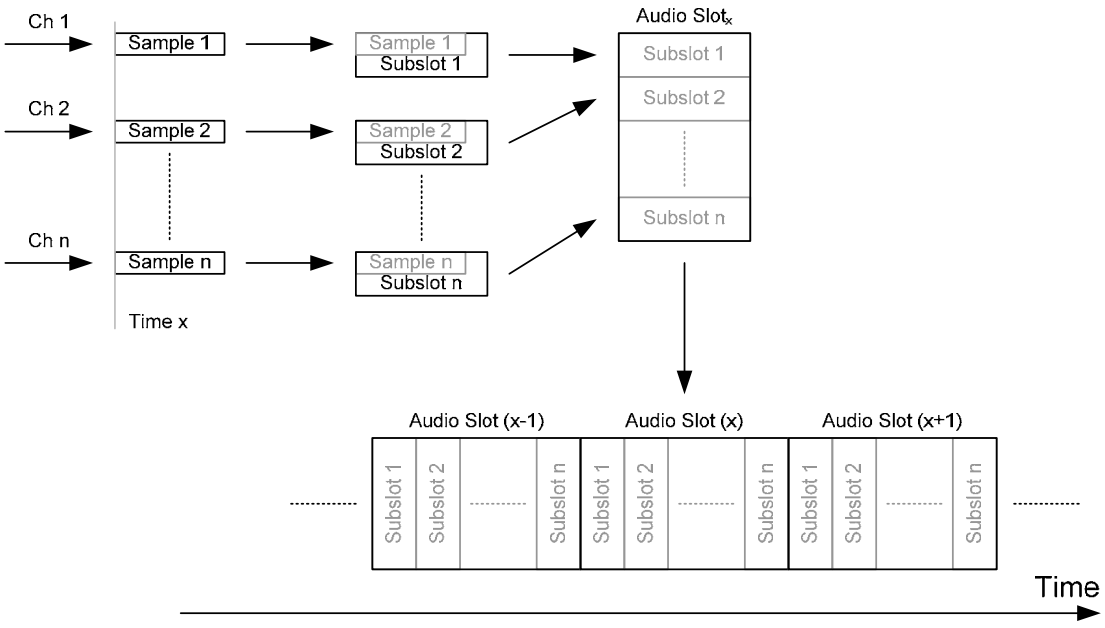

Figure 2-1: Type I Audio Stream

The second group, Type II, deals with those formats that do not preserve the notion of physical channels during the transmission over USB. Typically, all non-PCM encoded audio data streams belong to this group. A number of audio samples, often originating from multiple physical channels and taken over a certain period of time, are encoded into a number of bits in such a way that, after transmission, the original audio samples can be reconstructed to a certain degree of accuracy. The number of bits used for transmission is typically one or more orders of magnitude smaller than the number of bits needed to represent the original PCM audio samples, effectively realizing a considerable bandwidth reduction during transmission.

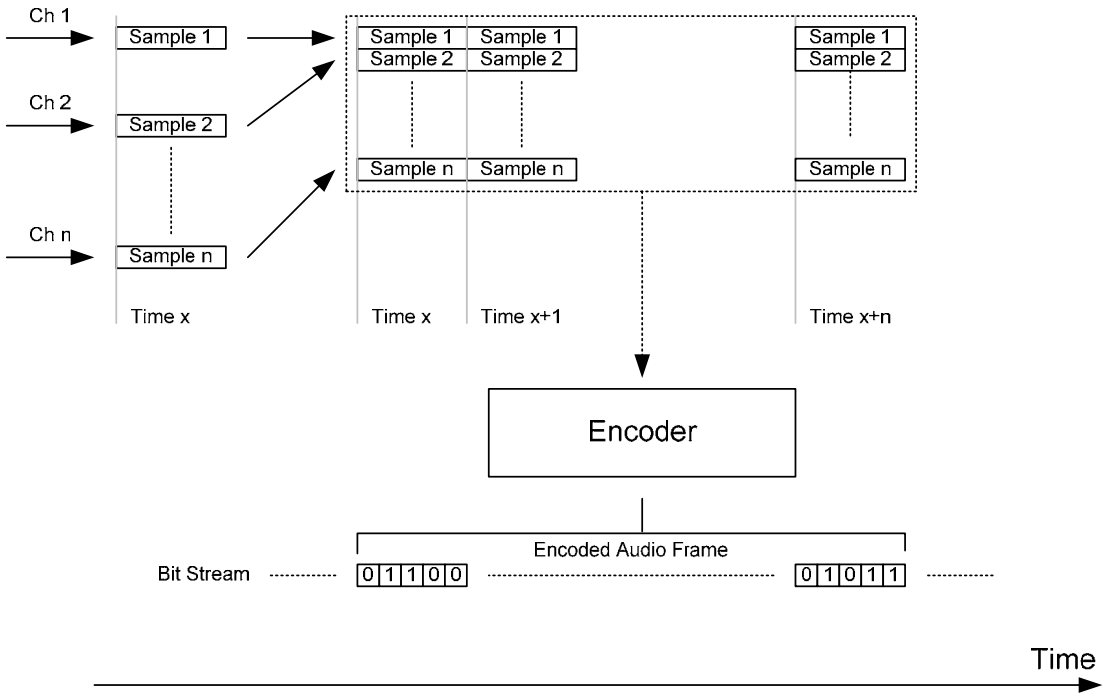

Figure 2-2: Type II Audio Stream

The third group, Type III, contains special formats that do not fit in both previous groups. In fact, they mix characteristics of Type I and Type II groups to transmit audio data streams 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 matches the sampling frequency of the original PCM audio data streams. Therefore, clock recovery at the receiving end is easier than it is in the case of Type II formats. The drawback is that unless multiple non-PCM encoded streams are packed into one pseudo stereo stream, more bandwidth than necessary is consumed.

The fourth 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) but 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. The capabilities of the AC-3 decoder are advertised by means of the AC-3 Decoder descriptor and the decoder Controls can be programmed through the AudioStreaming interface.

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, II, 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.

Section A.1, "Format Type Codes" summarizes the Audio Data Formats that are currently supported by the Audio Device Class. The following sections explain those formats 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 Virtual Frame and Virtual Frame Packet Definitions

To better describe packetization for audio the concept of a "virtual frame" (VF) is introduced. A virtual frame is defined as:

$$VF = (micro)frame * 2^{(bInterval-1)}$$

In addition, a "virtual frame packet" (VFP) is introduced. A virtual frame packet is defined as a packet that contains all the samples that are transferred over the bus during a virtual frame. For full-/high-speed endpoints, the virtual frame packets are exactly the same as the physical packets that are transferred over USB. However, for high-speed high-bandwidth endpoints, the virtual frame packet is the concatenation of the two or three physical packets that are transferred over the bus in a microframe.

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. See Section 5.12.3, "Clock Synchronization"

The above definitions provide a model of 'one (virtual frame) packet per (virtual) frame', irrespective of the actual transactions on the USB.

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 must 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.

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

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

where T_{VF} is the duration of a virtual frame 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 virtual frame packets must either contain INT(n_{av}) audio slots (small VFP) or INT(n_{av}) + 1 (large VFP) audio slots. For all *i*:

$$n_i = \text{INT}(n_{av}) \mid \text{INT}(n_{av}) + 1$$

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

Furthermore, a large VFP must be generated as soon as it becomes available. Typically, a source will generate a number of small VFPs as long as the accumulated fractional part of n_{av} remains < 1. Once the

accumulated fractional part of n_{av} becomes ≥ 1, the source must send a large VFP 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 VFP)/(large VFP) pattern generated by the source may be different from what the sink expects. Therefore, the sink must be capable to accept a large VFP 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 VF, it will send small VFPs of 44 audio slots. Each VF, it therefore sends '0.1 slot' too few and it will accumulate this fractional part in an accumulator. After having sent 9 small VFPs of 44 audio slots, at the tenth VF it will have exactly one audio slot in excess and therefore can send a large VFP 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 VFPs of 44 audio slots followed by 1 large VFP of 45 audio slots. The following table illustrates the process:

Table 2-1: Packetization

#VF	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 virtual frame. For all *i*:

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

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 or 4 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 must 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 zeros 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 must 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 virtual frame packets 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 VFP, a Transfer Delimiter must be sent instead.

2.3.1.6 Type I Format Type Descriptor

The Type I format type descriptor starts with the usual three fields: **bLength**, **bDescriptorType**, and **bDescriptorSubtype**.

The **bFormatType** field indicates this is a Type I descriptor. The **bSubslotSize** field indicates how many bytes are used to transport an audio subslot. The **bBitResolution** field indicates how many bits of the total number of available bits in the audio subslot are truly used by the audio function to convey audio information.

Table 2-2: Type I Format Type Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	Number	Size of this descriptor, in bytes: 6
1	bDescriptorType	1	Constant	CS_INTERFACE descriptor type.
2	bDescriptorSubtype	1	Constant	FORMAT_TYPE descriptor subtype.
3	bFormatType	1	Constant	FORMAT_TYPE_I. Constant identifying the Format Type the AudioStreaming interface is using.
4	bSubslotSize	1	Number	The number of bytes occupied by one audio subslot. Can be 1, 2, 3 or 4.

Offset	Field	Size	Value	Description
5	bBitResolution	1	Number	The number of effectively used bits from the available bits in an audio subslot.

2.3.1.7 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.2.1, "Audio Data Format Type I Bit Allocations."

2.3.1.7.1 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 zeros 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).

2.3.1.7.2 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].

2.3.1.7.3 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).

2.3.1.7.4 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.

2.3.1.7.5 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 II Formats

Type II formats are used to transmit non-PCM encoded audio data into bit streams that consist of a sequence of encoded audio frames.

2.3.2.1 Encoded Audio Frames

An encoded audio frame is a sequence of bits that contains an encoded representation of one or more physical audio channels. The encoding takes place over a fixed number of audio slots. Each encoded audio frame contains enough information to entirely reconstruct the audio samples (albeit not lossless), encoded in the encoded audio frame. No information from adjacent encoded audio frames is needed during decoding. The number of audio slots used to construct one encoded audio frame depends on the encoding scheme. (For MPEG, the number of slots per encoded audio frame (n_f) is 384 for Layer I or 1152 for Layer II. For AC-3, the number of slots is 1536.)

In most cases, the encoded audio frame represents multiple physical audio channels. The number of bits per encoded audio frame may be variable. The content of the encoded audio frame is defined according to the implemented encoding scheme. Where applicable, the bit ordering shall be MSB first, relative to existing standards of serial transmission or storage of that encoding scheme. An encoded audio frame represents an interval longer than the USB (micro)frame. This is typical of audio compression algorithms that use psycho-acoustic or vocal tract parametric models.

Note: It is important to make a clear distinction between a USB frame and an encoded audio frame. The overloaded use of the term frame could cause confusion. Therefore, this specification will always use the qualifier ‘encoded audio’ to refer to MPEG or AC-3 encoded audio frames.

2.3.2.2 Audio Bit Streams

An encoded audio bit stream is a concatenation of a potentially very large number of encoded audio frames, ordered according to ascending time. Subsequent encoded audio frames are independent and can be decoded separately.

2.3.2.3 USB Packets

Encoded audio bit streams are packetized when transported over an isochronous pipe. Each virtual frame packet potentially contains only part of a single encoded audio frame. Packet sizes are determined according to the short-packet protocol. The encoded audio frame is broken down into a number of packets, each containing **wMaxPacketSize** bytes except for the last packet, which may be smaller and contains the remainder of the encoded audio frame. If the **MaxPacketsOnly** bit D7 in the **bmAttributes** field of the class-specific endpoint descriptor is set, the last (short) packet must be padded with zero bytes to **wMaxPacketSize** length. No virtual frame packet may contain bits belonging to different encoded audio frames. If the encoded audio frame length is not a multiple of 8 bits, the last byte in the last packet is padded with zero bits. The decoder must ignore all padded extra bits and bytes. Consecutive encoded audio frames are separated by at least one Transfer Delimiter. A Transfer Delimiter must be sent in all virtual frames until the next encoded audio frame is due. The above rules guarantee that a new encoded audio frame always starts on a virtual frame packet boundary.

2.3.2.4 Bandwidth Allocation

The encoded audio frame time t_f equals the number of audio slots per encoded audio frame n_f divided by the sampling rate f_s of the original audio samples.

$$t_f = \frac{n_f}{f_s}$$

The allocated bandwidth for the pipe must accommodate for the largest possible encoded audio frame to be transmitted within an encoded audio frame time. This should take into account the Transfer Delimiter requirement and any differences between the time base of the stream and the USB (micro)frame timer. The device may choose to consume more bandwidth than necessary (by increasing the reported **wMaxPacketSize**) to minimize the time needed to transmit an entire encoded audio frame. This can be used to enable early decoding and therefore minimize system latency.

2.3.2.5 Timing

The timing reference point is the beginning of an encoded audio frame. Therefore, the USB packet that contains the first bits (usually the encoded audio frame sync word) of the encoded audio frame is used as a timing reference in USB space. This USB packet is called the reference packet. The transmission of the reference packet of an encoded audio frame should begin at the target playback time of that frame (minus the endpoint's reported delay) rounded to the nearest USB (micro)frame time. This guarantees that, at the receiving end, the arrival of subsequent reference packets matches the encoded audio frame time t_f as closely as possible.

2.3.2.6 Type II Format Type Descriptor

The Type II Format Type descriptor starts with the usual three fields **bLength**, **bDescriptorType** and **bDescriptorSubtype**.

The **bFormatType** field indicates this is a Type II descriptor. The **wMaxBitRate** field contains the maximum number of bits per second this interface can handle. It is a measure for the buffer size available in the interface. The **wSlotsPerFrame** field contains the number of PCM audio slots contained within a single encoded audio frame.

Table 2-3: Type II Format Type Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	Number	Size of this descriptor, in bytes: 8
1	bDescriptorType	1	Constant	CS_INTERFACE descriptor type.
2	bDescriptorSubtype	1	Constant	FORMAT_TYPE descriptor subtype.
3	bFormatType	1	Constant	FORMAT_TYPE_II. Constant identifying the Format Type the AudioStreaming interface is using.
4	wMaxBitRate	2	Number	Indicates the maximum number of bits per second this interface can handle. Expressed in kbits/s.
6	wSlotsPerFrame	2	Number	Indicates the number of PCM audio slots contained in one encoded audio frame.

2.3.2.7 Rate feedback

If the isochronous data endpoint needs explicit rate feedback (adaptive source, asynchronous sink), the feedback pipe must report the number of equivalent PCM audio slots. The host will accumulate this data and start transmission of an encoded audio frame whenever the current number of audio slots exceeds the number of slots per encoded audio frame. The remainder is kept in the accumulator.

2.3.2.8 Type II Supported Formats

The following sections list all currently supported Type II Audio Data Formats. The bit allocations in the **bmFormats** field of the class-specific AS interface descriptor for the different Type II Audio Data Formats can be found in Appendix A.2.2, "Audio Data Format Type II Bit Allocations."

2.3.2.8.1 MPEG Format

Refer to the ISO/IEC 11172-3:1993 "Information technology -- Coding of moving pictures and associated audio for digital storage media at up to about 1,5 Mbit/s -- Part 3: Audio" and the ISO/IEC 13818-3:1998 "Information technology -- Generic coding of moving pictures and associated audio information -- Part 3: Audio" specifications for detailed format information.

2.3.2.8.2 AC-3 Format

Refer to the Digital Audio Compression Standard (AC-3), ATSC A/52A Aug. 20, 2001 for detailed format information.

2.3.2.8.3 WMA Format

This is an audio compression format from Microsoft. For technical and licensing information, contact Microsoft directly (http://www.microsoft.com/windows/windowsmedia/default.aspx).

2.3.2.8.4 DTS Format

Refer to the ETSI Specification TS 102 114, "DTS Coherent Acoustics; Core and Extensions". Available from http://webapp.etsi.org/action%5CPU/20020827/ts_102114v010101p.pdf.

2.3.2.8.5 Type II Raw Data

This audio format is included to allow transport of data (audio or other) over a USB AudioStreaming interface in the form of a bit stream 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 II formatted audio data, but the data is never to be interpreted as being audio data.

2.3.3 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 must 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.

2.3.3.1 Type III Format Type Descriptor

The **bFormatType** field indicates this is a Type III descriptor. The **bSubSlotSize** field indicates how many bytes are used to transport an audio subslot. The **bBitResolution** field indicates how many bits of the total number of available bits in the audio subslot are truly used by the audio function to convey audio information.

Table 2-4: Type III Format Type Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	Number	Size of this descriptor, in bytes: 6
1	bDescriptorType	1	Constant	CS_INTERFACE descriptor type.
2	bDescriptorSubtype	1	Constant	FORMAT_TYPE descriptor subtype.
3	bFormatType	1	Constant	FORMAT_TYPE_III. Constant identifying the Format Type the AudioStreaming interface is using.
4	bSubslotSize	1	Number	The number of bytes occupied by one audio subslot. Must be set to two.
5	bBitResolution	1	Number	The number of effectively used bits from the available bits in an audio subframe.

2.3.3.2 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.2.3, “Audio Data Format Type III Bit Allocations.”

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

- IEC61937_AC-3
- IEC61937_MPEG-1_Layer1
- IEC61937_MPEG-1_Layer2/3 or IEC61937_MPEG-2_NOEXT
- IEC61937_MPEG-2_EXT
- IEC61937_MPEG-2_AAC_ADTS
- IEC61937_MPEG-2_Layer1_LS
- IEC61937_MPEG-2_Layer2/3_LS
- IEC61937_DTS-I
- IEC61937_DTS-II
- IEC61937_DTS-III
- IEC61937_ATRAC
- IEC61937_ATRAC2/3

In addition, the WMA audio compression format as defined by Microsoft is supported.

2.3.4 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 control an encoder or decoder process in one or more of its Alternate Settings. 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).

2.3.4.1 Type IV Format Type Descriptor

The **bFormatType** field indicates this is a Type IV descriptor.

Table 2-5: Type IV Format Type Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	Number	Size of this descriptor, in bytes: 4
1	bDescriptorType	1	Constant	CS_INTERFACE descriptor type.
2	bDescriptorSubtype	1	Constant	FORMAT_TYPE descriptor subtype.
3	bFormatType	1	Constant	FORMAT_TYPE_IV. Constant identifying the Format Type the AudioStreaming interface is using.

2.3.4.2 Type IV Supported Formats

This specification supports all Audio Data Formats on an external connection that are defined on a USB pipe (Type I, II, and III). See Section 2.3.1.7, “Type I Supported Formats”, Section 2.3.2.8, “Type II Supported Formats”, and Section 2.3.3.2, “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.2.4, “Audio Data Format Type IV Bit Allocations.”

2.4 Extended Audio Data Formats

Extended Audio Data Formats add support for a Packet Header to the previously defined Simple Audio Data Formats Type I, II, and III. For the Extended Audio Data Format Type I, an additional optional synchronous Control Channel is defined.

2.4.1 Extended Type I Formats

Extended Audio Data Format Type I adds support for both a Packet Header and a synchronous Control Channel to the Simple Type I Format definition.

All three elements (Packet Header, audio data, and Control Channel) of an Extended Type I packet are optional. The Extended Format Type I descriptor (see further) indicates which elements are present. It is therefore possible to provide only a Control Channel, without Packet Header or audio data. The following figure further illustrates the concept.

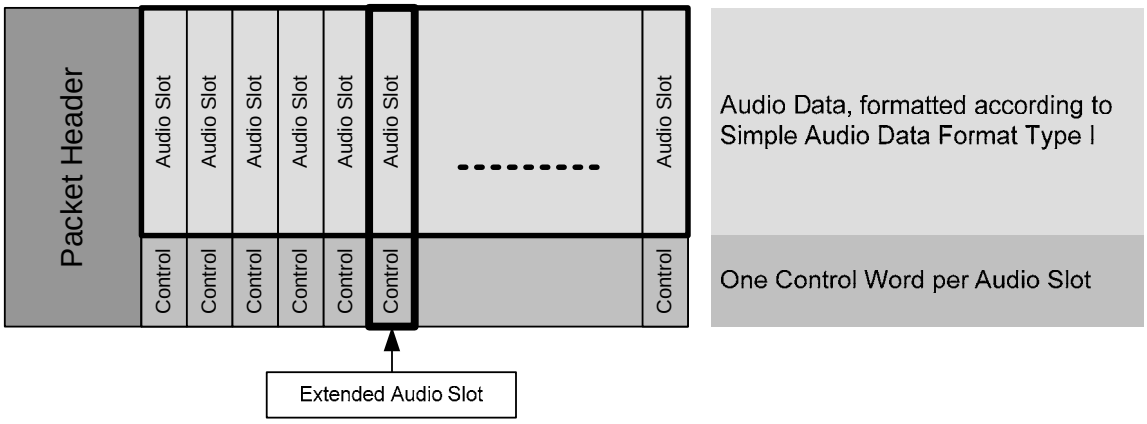

Figure 2-3: Extended Type I Format

Each Virtual Frame Packet (VFP) can start with an optional Packet Header. If Packet Headers are used, they must be present in every VFP. The length of the Packet Header must be the same for every VFP. The Packet Header is then followed by a number of Extended Audio Slots. An Extended Audio Slot is the concatenation of a Control Word, followed by the Type I Audio Slot. The Control Channel therefore consists of a stream of Control Words, where each Control Word is synchronous to its associated Audio Slot. There are as many Control Channel Words per VFP as there are Audio Slots in the VFP. The byte size of the Control Words is independent of the Audio Subslot size and is the same for each Audio Slot.

2.4.1.1 Extended Type I Format Type Descriptor

The first part of the Extended Type I Format Type descriptor is identical to the Simple Type I Format Type descriptor (See Section 2.3.1.6, "Type I Format Type Descriptor".) Three additional fields are added to describe the Packet Header and the Control Channel.

The **bHeaderLength** field indicates the number of bytes contained in the Packet Header. The **bControlSize** field indicates the size in bytes of each Control Channel Word in the stream. The **bSideBandProtocol** field contains a constant identifying the Side Band Protocol that is used for the Packet Header and Control Channel. This specification defines a number of Side Band Protocols (See Section 2.4.4, "Side Band Protocols").

If the Packet Header is not used, then the **bHeaderLength** field must be set to 0. Likewise, if the Control Channel is not implemented, then the **bControlSize** field must be set to 0. 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) must all be set to 0.

Table 2-6: Extended Type I Format Type Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	Number	Size of this descriptor, in bytes: 9
1	bDescriptorType	1	Constant	CS_INTERFACE descriptor type.
2	bDescriptorSubtype	1	Constant	FORMAT_TYPE descriptor subtype.
3	bFormatType	1	Constant	EXT_FORMAT_TYPE_I. Constant identifying the Format Type the AudioStreaming interface is using.
4	bSubslotSize	1	Number	The number of bytes occupied by one audio subslot. Can be 1, 2, 3 or 4.

Offset	Field	Size	Value	Description
5	bBitResolution	1	Number	The number of effectively used bits from the available bits in an audio subslot.
6	bHeaderLength	1	Number	Size of the Packet Header, in bytes.
7	bControlSize	1	Number	Size of the Control Channel Words, in bytes.
8	bSideBandProtocol	1	Constant	Constant, identifying the Side Band Protocol used for the Packet Header and Control Channel content.

2.4.2 Extended Type II Formats

Extended Audio Data Format Type II adds support for a Packet Header to the Simple Type II Format definition.

The elements (Packet Header and audio data) of an Extended Type II packet are optional. The Extended Format Type II descriptor (see further) indicates which elements are present. It is therefore possible to provide only a Packet Header without audio data. The following figure further illustrates the concept.

Figure 2-4: Extended Type II Format

Each Virtual Frame Packet (VFP) can start with an optional Packet Header. If Packet Headers are used, they must be present in every VFP. The length of the Packet Header must be the same for every VFP. The Packet Header is then followed by the actual encoded audio frame data.

2.4.2.1 Extended Type II Format Type Descriptor

The first part of the Extended Type II Format Type descriptor is identical to the Simple Type II Format Type descriptor (See Section 2.3.2.6, “Type II Format Type Descriptor”.) Two additional fields are added to describe the Packet Header.

The **bHeaderLength** field indicates the number of bytes contained in the Packet Header. The **bSideBandProtocol** field contains a constant identifying the Side Band Protocol that is used for the Packet Header. This specification defines a number of Side Band Protocols (See Section 2.4.4, “Side Band Protocols”).

If the Packet Header is not used, then the **bHeaderLength** field must be set to 0. 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) must all be set to 0.

Table 2-7: Extended Type II Format Type Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	Number	Size of this descriptor, in bytes: 10

Offset	Field	Size	Value	Description
1	bDescriptorType	1	Constant	CS_INTERFACE descriptor type.
2	bDescriptorSubtype	1	Constant	FORMAT_TYPE descriptor subtype.
3	bFormatType	1	Constant	Ext_FORMAT_TYPE_II. Constant identifying the Format Type the AudioStreaming interface is using.
4	wMaxBitRate	2	Number	Indicates the maximum number of bits per second this interface can handle. Expressed in kbits/s.
6	wSamplesPerFrame	2	Number	Indicates the number of PCM audio samples contained in one encoded audio frame.
8	bHeaderLength	1	Number	Size of the Packet Header, in bytes.
9	bSideBandProtocol	1	Constant	Constant, identifying the Side Band Protocol used for the Packet Header content.

2.4.3 Extended Type III Formats

Extended Audio Data Format Type III adds support for a Packet Header to the Simple Type III Format definition.

The elements (Packet Header and audio data) of an Extended Type III packet are optional. The Extended Format Type III descriptor (see further) indicates which elements are present. It is therefore possible to provide only a Packet Header without audio data. The following figure further illustrates the concept.

Figure 2-5: Extended Type III Format

Each Virtual Frame Packet (VFP) can start with an optional Packet Header. If Packet Headers are used, they must be present in every VFP. The length of the Packet Header must be the same for every VFP. The Packet Header is then followed by the actual encoded audio frame data.

2.4.3.1 Extended Type III Format Type Descriptor

The first part of the Extended Type III Format Type descriptor is identical to the Simple Type III Format Type descriptor (See Section 2.3.3.1, “Type III Format Type Descriptor”.) Two additional fields are added to describe the Packet Header.

The **bHeaderLength** field indicates the number of bytes contained in the Packet Header. The **bSideBandProtocol** field contains a constant identifying the Side Band Protocol that is used for the Packet Header. This specification defines a number of Side Band Protocols (See Section 2.4.4, “Side Band Protocols”).

If the Packet Header is not used, then the **bHeaderLength** field must be set to 0. 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) must all be set to 0.

Table 2-8: Extended Type III Format Type Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	Number	Size of this descriptor, in bytes: 8
1	bDescriptorType	1	Constant	CS_INTERFACE descriptor type.
2	bDescriptorSubtype	1	Constant	FORMAT_TYPE descriptor subtype.
3	bFormatType	1	Constant	EXT_FORMAT_TYPE_III. Constant identifying the Format Type the AudioStreaming interface is using.
4	bSubslotSize	1	Number	The number of bytes occupied by one audio subslot. Must be set to two.
5	bBitResolution	1	Number	The number of effectively used bits from the available bits in an audio subslot.
6	bHeaderLength	1	Number	Size of the Packet Header, in bytes.
7	bSideBandProtocol	1	Constant	Constant, identifying the Side Band Protocol used for the Packet Header content.

2.4.4 Side Band Protocols

This specification currently defines a single Side Band Protocol. Additional Protocols can be added later if needed.

2.4.4.1 Presentation Timestamp Side Band Protocol

The Presentation Timestamp protocol only uses the Packet Header to convey high resolution time information over the isochronous pipe. The Packet header is 12 bytes in size. It must occur at the start of each VFP.

Bit D0 in the **bmFlags** field indicates whether this is a valid timestamp (D0 = 0b1) or a repeated or non-valid timestamp (D0 = 0b0). When D0 is set to zero, the time fields of the Packet Header must be ignored.

The **qNanoSeconds** field indicates the time T at which the first sample in the VFP needs to be rendered with respect to the start of the stream (T = 0). The qNanoSeconds field can range from 0 to 2^{63}-1 ns (Bit 63 is considered to be a sign bit and must be set to zero). It is up to the entity that generates the timestamp to decide to which accuracy the timestamp will be rendered.

Table 2-9: Hi-Res Presentation TimeStamp Layout

Offset	Field	Size	Value	Description
0	bmFlags	4	Bitmap	D30..0: Reserved. Must be set to 0. D31: Valid.

Offset	Field	Size	Value	Description
4	qNanoSeconds	8	Number	Offset in nanoseconds from the beginning of the audio stream.

Note: Timing information is intrinsically provided by the isochronous data transport mechanism itself (packets are synchronized to the USB SOF and the number of samples per packet is an overall measure of the audio data sampling rate). However, the high resolution presentation timestamp could potentially be used to deliver more accurate instantaneous timing information to the sink or to report a (constant) delay between the moment of transport over the USB and the moment of actual rendition. Care must be taken to ensure that the information contained in the Packet Header is at all times in agreement with the implicit timing information, delivered by the isochronous streaming mechanism.

3 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.

4 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 Format Type Codes

Table A-1: Format Type Codes

Format Type Code	Value
FORMAT_TYPE_UNDEFINED	0x00
FORMAT_TYPE_I	0x01
FORMAT_TYPE_II	0x02
FORMAT_TYPE_III	0x03
FORMAT_TYPE_IV	0x04
EXT_FORMAT_TYPE_I	0x81
EXT_FORMAT_TYPE_II	0x82
EXT_FORMAT_TYPE_III	0x83

A.2 Audio Data Format Bit Allocation in the bmFormats field

A.2.1 Audio Data Format Type I Bit Allocations

Table A-2: Audio Data Format Type I Bit Allocations

Name	bmFormats
PCM	D0
PCM8	D1
IEEE_FLOAT	D2
ALAW	D3
MULAW	D4
Reserved. Must be set to 0.	D30..D5
TYPE_I_RAW_DATA	D31

A.2.2 Audio Data Format Type II Bit Allocations

Table A-3: Audio Data Format Type II Bit Allocations

Name	bmFormats
MPEG	D0

Name	bmFormats
AC-3	D1
WMA	D2
DTS	D3
Reserved. Must be set to 0.	D30..D4
TYPE_II_RAW_DATA	D31

A.2.3 Audio Data Format Type III Bit Allocations

Table A-4: Audio Data Format Type III Bit Allocations

Name	bmFormats
IEC61937_AC-3	D0
IEC61937_MPEG-1_Layer1	D1
IEC61937_MPEG-1_Layer2/3 or IEC61937_MPEG-2_NOEXT	D2
IEC61937_MPEG-2_EXT	D3
IEC61937_MPEG-2_AAC_ADTS	D4
IEC61937_MPEG-2_Layer1_LS	D5
IEC61937_MPEG-2_Layer2/3_LS	D6
IEC61937_DTS-I	D7
IEC61937_DTS-II	D8
IEC61937_DTS-III	D9
IEC61937_ATRAC	D10
IEC61937_ATRAC2/3	D11
TYPE_III_WMA	D12
Reserved. Must be set to 0.	D31..D13

A.2.4 Audio Data Format Type IV Bit Allocations

Table A-5: Audio Data Format Type IV Bit Allocations

Name	bmFormats
PCM	D0

Name	bmFormats
PCM8	D1
IEEE_FLOAT	D2
ALAW	D3
MULAW	D4
MPEG	D5
AC-3	D6
WMA	D7
IEC61937_AC-3	D8
IEC61937_MPEG-1_Layer1	D9
IEC61937_MPEG-1_Layer2/3 or IEC61937_MPEG-2_NOEXT	D10
IEC61937_MPEG-2_EXT	D11
IEC61937_MPEG-2_AAC_ADTS	D12
IEC61937_MPEG-2_Layer1_LS	D13
IEC61937_MPEG-2_Layer2/3_LS	D14
IEC61937_DTS-I	D15
IEC61937_DTS-II	D16
IEC61937_DTS-III	D17
IEC61937_ATRAC	D18
IEC61937_ATRAC2/3	D19
TYPE_III_WMA	D20
IEC60958_PCM	D21
Reserved. Must be set to 0.	D31..D22

A.3 Side Band Protocol Codes

Table A-6: Side Band Protocol Codes

Protocol Code	Value
PROTOCOL_UNDEFINED	0x00

Protocol Code	Value
PRES_TIMESTAMP_PROTOCOL	0x01