Universal Serial Bus
Device Class Definition
for
Audio/Video Devices

(Developed pursuant to USB-IF IP agreement for Video Display and subsequently renamed as above)

AVFormat 2 – Isochronous Audio

Release 1.0

December 07, 2011

Scope of This Release

This document is the Release 1.0 of this AVFormat 2 – Isochronous Audio Definition.

Contributors

Jim Hunkins	AMD
Jason Hawken	AMD
Kenneth Ma	Broadcom
Hans van Antwerpen	Cypress Semiconductor
Jitendra Kulkarni	Cypress Semiconductor
Dan Ellis	DisplayLink
Trevor Hall	DisplayLink
Alec Cawley	DisplayLink
David Dolby	Dolby Labs
David Roh	Dolby Labs
Tsehao Lee	Grain Media
Pierre Bossart	Intel
David Harriman	Intel
Abdul R. Ismail (Chair)	Intel
J.P. Giacalone	Intel
Steve McGowan	Intel
Sridharan Ranganathan	Intel
Yoav Nissim	Jungo
Ygal Blum	Jungo
Max Basler	Littelfuse
Paul E. Berg	MCCI
Cristian Chis	MCCI
John Garney	MCCI
Geert Knapen (Editor)	MCCI
Tomi Heinonen	Nokia Corporation
Richard Petrie	Nokia Corporation
Yoram Rimoni	Qualcomm, Inc.
Mark Bohm	SMSC
John Sisto	SMSC
Morgan Monks	SMSC
Bruno Paillard	Soft-dB

Will Harris Texas Instruments
Grant Ley Texas Instruments

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.

All product names are trademarks, registered trademarks, or service marks of their respective owners.

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

Table of Contents

List of Tables

List of Figures

1. Introduction

1.1. Scope

This document describes in detail all the Audio-Only AV Data Formats that are supported by the AV Device Class. This document is considered an integral part of the AV Device Class Definition, although subsequent revisions of this document are independent of the revision evolution of the main USB AV Device Class Definition. This is to easily accommodate the addition of new Audio-Only Data Formats without requiring changes to other USB AV documents.

1.2. Purpose

The purpose of this document is to provide all necessary information a designer needs to build AVFunctions that use the Audio Formats described here on at least one of its AVData Audio Streaming interfaces.

1.3. Related Documents

- [USB2.0] – Universal Serial Bus Specification, Revision 2.0, April 27, 2000 (referred to in this document as the USB 2.0 Specification) (available at: http://www.usb.org/developers/docs).
- [USB3.0] – Universal Serial Bus 3.0 Specification, Revision 1.0 (including errata and ECN's through May 1, 2011), June 6, 2011 (referred to in this document as the USB 3.0 Specification) (available at: http://www.usb.org/developers/docs.)
- [AUDIO1.0] – Universal Serial Bus Device Class Definition for Audio Devices, Release 1.0, March 18, 1998 (available at: http://www.usb.org/developers/devclass_docs).
- [FORMATS1.0] – Universal Serial Bus Device Class Definition for Audio Data Formats, Release 1.0, March 18, 1998 (available at: http://www.usb.org/developers/devclass_docs).
- [TERMTYPES1.0] – Universal Serial Bus Device Class Definition for Terminal Types, Release 1.0, March 18, 1998 (available at: http://www.usb.org/developers/devclass_docs).
- [AUDIO2.0] – Universal Serial Bus Device Class Definition for Audio Devices, Release 2.0, May 31, 2006 (available at: http://www.usb.org/developers/devclass_docs).
- [FORMATS2.0] – Universal Serial Bus Device Class Definition for Audio Data Formats, Release 2.0, May 31, 2006 (available at: http://www.usb.org/developers/devclass_docs).
- [TERMTYPES2.0] – Universal Serial Bus Device Class Definition for Terminal Types, Release 2.0, May 31, 2006 (available at: http://www.usb.org/developers/devclass_docs).
- [USBCS] – Universal Serial Bus Device Class Definition for Content Security Devices – Content Security Framework, Revision 2.0 (available at: http://www.usb.org/developers/devclass_docs).
- [USBCSM-5] – Universal Serial Bus Device Class Definition for Content Security Devices – Content Security Method 5 – High-bandwidth Digital Content Protection 2.1 (HDCP 2.1) Implementation, Revision 1.0 (available at: http://www.usb.org/developers/devclass_docs).
- USBECNIAD – USB Engineering Change Notice: Interface Association Descriptors (available at: http://www.usb.org/developers/docs).
- [USBIADDCC] – USB Interface Association Descriptor Device Class Code and Use Model, Revision 1.0, July 23, 2003 (available at: http://www.usb.org/developers/whitepapers).
- [USBLANGIDS] – Universal Serial Bus Language Identifiers (LANGIDs), Revision 1.0, March 29, 2000 (available at: http://www.usb.org/developers/docs).
- [AVFUNCTION] – Universal Serial Bus Device Class Definition for Audio/Video Devices – AV Device Class Overview & AVFunction Definition, Release 1.0, December 07, 2011 (available at: http://www.usb.org/developers/devclass_docs).
- [AVFORMAT_1] – Universal Serial Bus Device Class Definition for Audio/Video Devices – AVFormat 1 – Video over Bulk, Release 1.0, December 07, 2011 (available at: http://www.usb.org/developers/devclass_docs).
- [AVFORMAT_2] – Universal Serial Bus Device Class Definition for Audio/Video Devices – AVFormat 2 – Isochronous Audio, Release 1.0, December 07, 2011 (available at: http://www.usb.org/developers/devclass_docs).

- [AVFORMAT_3] – Universal Serial Bus Device Class Definition for Audio/Video Devices – AVFormat 3 – Uncompressed Full Frame Isochronous Video, Release 1.0, December 07, 2011 (available at: http://www.usb.org/developers/devclass_docs).
- [AVSCHEMA] – Available at: http://avschemas.usb.org/v1/avschema.xsd
- [BDP] – Universal Serial Bus Device Class Definition for Audio/Video Devices – Basic Device Profile, Release 1.0 (available at: http://www.usb.org/developers/whitepapers).
- [ANSIS1_11] – ANSI S1.11-2004 (R2009) standard (available at: http://www.ansi.org).
- [IEC11172_3] – MPEG-1 standard 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 (available at http://www.iec.ch).
- [IEC13818_1] – MPEG-2 standard ISO/IEC 13818:2000 Information technology – Generic coding of moving pictures and associated audio information – Part 1: Systems (available at http://www.iec.ch).
- [IEC13818_3] – MPEG-2 standard ISO/IEC 13818:1998 Information technology – Generic coding of moving pictures and associated audio information – Part 3: Audio" (available at http://www.iec.ch).
- [AC_3] – Digital Audio Compression Standard (AC-3, Enhanced AC-3), ETSI TS 102 366 (available at http://www.etsi.org).
- [IEEE_754] – ANSI/IEEE-754 floating-point standard (available at http://www.ieee.org).
- [IEC60958] – ISO/IEC 60958 International Standard: Digital Audio Interface and Annexes (available at: http://www.iec.ch).
- [IEC61937] – ISO/IEC 61937 standard (available at: http://www.iec.ch).
- [ETSI_TS_102_114] – ETSI Specification TS 102 114, "DTS Coherent Acoustics; Core and Extensions" (available at http://www.etsi.org).
- [HDCP2.1] – High-bandwidth Digital Content Protection System. Interface Independent Adaptation. Revision 2.1, July 18, 2011 (available from http://www.digital-cp.com).
- [MLP] – DVD Specifications for High Definition Video: MLP Reference Information.
- [IEC14496_3] – MPEG-4 Standard ISO/IEC 14496-3 – Information Technology – Coding of audio-visual objects – Part 3: Audio (available at http://www.iec.ch).
- [IEC14496_10] – MPEG-4 Standard ITU-T H.264 and ISO/IEC 14496-10:2004 – Information Technology – Coding of audio-visual objects – Part 10: Advanced Video Coding. Second Edition 2004-10-01 (available at http://www.iec.ch).
- [WMA] – Audio compression format from Microsoft. For technical and licensing information, contact Microsoft directly (http://www.microsoft.com/windows/windowsmedia/default.aspx).
- [HDMI] – The official High Definition Multimedia Interface website http://www.hdmi.org.
- [RFC5646] – Tags for Identifying Languages, September 2009 (available at http://www.rfc-editor.org/rfc/rfc5646.txt).
- [KHRONOS] – The Khronos Group, Open Standards for Media Authoring and Acceleration (available at http://www.khronos.org).
- [CEA-861-E] – A DTV Profile for Uncompressed High Speed Digital Interfaces, March 2008 (available at http://www.cea.org).
- [VESA] – Video Electronics Standards Association (available at http://www.vesa.org).
- [IEC10918_4] – JPEG Standard ISO/IEC 10918-4 – Information technology – Digital compression and coding of continuous-tone still images: Registration of JPEG profiles, SPIFF profiles, SPIFF tags, SPIFF colour spaces, APPn markers, SPIFF compression types and Registration Authorities (REGAUT). First Edition 1999-08-15 (available at http://www.iec.ch).

1.4. 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 [USB2.0] and [USB3.0].

Table 1-1: Terms and Abbreviations

Term	Description
2D Video	A VideoStream consisting of a single VideoChannel, providing a monoscopic video experience.

Term	Description
3D2 Video	A VideoStream consisting of 2 separate VideoChannels, providing a stereoscopic 3-dimensional video experience. One VideoChannel is intended for the left eye, and the other VideoChannel is intended for the right eye.
3Dn Video	A VideoStream consisting of n separate VideoChannels, providing a multiscopic 3-dimensional video experience. This type of VideoStream usually requires a sophisticated lenticular system in front of the display to ensure that the viewer only sees 2 (one for the left eye, one for the right eye) out of the n channels at the same time. Depending on the position of the viewer with respect to the display, a different set of 2 VideoChannels is made visible to the viewer.
AAC	Advanced Audio Coding.
AC-3	Audio compression standard from Dolby Labs.
AudioBundle	AudioChannels are physically organized into AudioTracks, and AudioTracks are then organized into the AudioBundle where each AudioChannel has an AudioChannel Type associated with it. Very similar to the AudioCluster concept, but the physical characteristics of the audio stream, such as AudioFrame Rate and bit resolution, are retained in the AudioBundle.
(Logical) AudioChannel	A logical transport medium for a single audio channel. Makes abstraction of the physical Properties and formats of the connection. Is identified by AudioChannel Type.
AudioChannel Type	The spatial location of an AudioChannel. Uniquely identifies the AudioChannel. Examples are Front Left channel (FL), Back Right channel (BR), etc.
AudioCluster	The group of all the audio-only logical AudioChannels in an AVCluster.
AudioControl Property	A parameter of an AudioControl. Examples are Current, Next, Range Properties of a Volume Control.
AudioControl	A logical object that is used to manipulate a specific audio Property. Examples are Volume Control, Mute Control, etc.
AudioSample	The basic representation of an audio signal (one channel), sampled (digitized) at a specific moment in time.
AudioSlot	A collection of AudioSubSlots, each containing an AudioSample of a different physical audio channel, taken at the same moment in time.
AudioStream	A concatenation of a potentially very large number of AudioSlots ordered according to ascending time where the AudioSamples are formatted according to one of the Audio Formats described in this specification and where the AudioChannels are organized in an AudioBundle.
AudioStreamConfig (AudioStream Configuration)	An XML Description that provides all the information necessary to fully characterize an AudioStream. Includes an AudioBundle, AudioFrame, and AudioSample component.
AudioStreamConfigList	A list of AudioStreamConfig Descriptions used to indicate the different AudioStream Configurations an AVData Audio Streaming Interface supports.
AudioSubSlot	Holds a single AudioSample of a single audio channel.
AV Description Document (AVDD)	An XML-formatted document that provides a complete description of all the AV class-specific elements and features of the AVFunction.
AV Interface Association	A grouping of a single AVControl Interface, and zero or more AVData Streaming Interfaces that together constitute a complete interface to an AVFunction.
AV Profile	A set of limitations and restricting rules, bundled in a single document that applies to the AV Device Class Definition. A Profile defines a specific type of AVFunction that is guaranteed to interoperate with all Controllers that support that Profile.
AVBundle	A group of physical VideoChannels AudioChannels and MetadataChannels that carry tightly related (from a content perspective) synchronous video, audio, and metadata information over a USB pipe.
AVCluster	A group of logical VideoChannels AudioChannels and MetadataChannels that carry tightly related (from a content perspective) synchronous video, audio, and metadata information over an interconnect within the AVCore.
AVCore	That part of the AVFunction that contains most of the building blocks (Terminals, Units) that makes up most of the AVControl functionality of the AVFunction. Explicitly excludes the AVData Streaming Interfaces that are part of the AVFunction.
AVControl Interface	A USB interface used to access the AVControls inside an AVFunction.
AVData Entity	An addressable Entity representing a source of AV data flowing into or a sink for AV data flowing out of the AVCore (including AVData Streaming Interfaces).
AVData Streaming Interface	A USB interface used to transport AVStreams into or out of the AVFunction.

Term	Description
AVFunction	An independent part of a USB device that deals with AV-related functionality. Includes the AVCore and all AVData Entities.
AVHeader	The Header present in the AudioPayload or VideoPayload of a Command or Notify Message. Also, the Header present in every SIP.
AVStream	A generic name for either a VideoStream or an AudioStream. See VideoStream and AudioStream.
AVStream Configuration	A generic name for either a VideoStream Configuration or an AudioStream Configuration. See VideoStream Configuration and AudioStream Configuration.
BDP	Basic Device Profile. See [BDP]
BIB	Bus Interval Boundary ([USB3.0]).
CBP	Control Bulk Pair. A pair of one Bulk IN and one Bulk OUT endpoint in the AVControl Interface that is used for AV class-specific messaging.
CBP Idle Condition	A condition when an AVFunction determines that it will not have data available on its CBP Bulk IN pipe for a certain amount of time (AV_CBP_IDLE_TIME) and informs the Host of this condition to avoid having the Host continually issue IN tokens for that pipe.
Channel	A logical channel in an AVCluster. Can be a VideoChannel, AudioChannel, or MetadataChannel.
Controllee	The entity that is controlled by the Controller. The AVFunction always assumes this role.
Controller	The entity that controls the AVFunction. In this version of the specification, the USB Host assumes this role.
Control Sequence	A sequence of a Command and Response Message that is used to convey or retrieve one or more control parameters to or from the AVFunction.
Converter Unit (CU)	Provides the means to transform an incoming VideoTrack and/or AudioTrack into another VideoTrack and/or AudioTrack with different characteristics.
CUD	Converter Unit Description.
Description	Section of the AVDD (AV Description Document) that provides a detailed XML description of a specific Entity or other building block of the AVFunction architecture. Replaces the concept of the class-specific USB Descriptor.
DTS	Digital Theater Systems.
DUD	Metadata Unit Description.
DVD	Digital Versatile Disc.
Effect Unit (EU)	Provides advanced AV manipulation on the incoming logical AudioChannels.
Encoded Audio Bit Stream	A concatenation of a potentially very large number of encoded audio frames, ordered according to ascending time.
Encoded AudioFrame	A sequence of bits that contains an encoded representation of AudioSamples from one or more physical audio channels taken over a fixed period of time.
Encoded VideoFrame	A sequence of bits that contains an encoded representation of VideoSamples from one VideoFrame.
Entity	An addressable logical object inside an AVFunction.
EUD	Effect Unit Description.
Feature Unit (FU)	Provides basic AV manipulation on the incoming logical AV channels.
FUD	Feature Unit Description.
GraphicsEngine Entity (GEE)	An addressable Entity inside the AVCore representing a graphics subsystem of the AVFunction.
H.264	ITU name of ISO MPEG4 part10 Advanced Video Coding standard. See [IEC14496_10].
HE_AAC	High Efficiency Advanced Audio Coding
Header	A collection of SubHeaders present in a SIP, containing Extended Audio Format data.
Input Pin	A logical input connection to an Entity. Carries a single AVCluster.
Input Terminal (IT)	A receptacle for AV information flowing into the AVFunction.
Inter-VideoFrame	A VideoFrame composed of Intra-coded and Inter-coded MacroBlocks.
Intra-VideoFrame	A VideoFrame composed of Intra-coded MacroBlocks only.

Term	Description
ITD	Input Terminal Description.
MacroBlock	An elementary group of VideoSamples considered by an H.264 encoder.
MetadataBundle	The group of all the metadata-only physical channels in an AVBundle.
(Logical) MetadataChannel	A logical transport medium for a single metadata channel. Makes abstraction of the physical Properties and formats of the connection. Is identified by MetadataChannel Type.
MetadataChannel Type	The metadata type of a MetadataChannel. Uniquely identifies the MetadataChannel. Examples are Subtitle (SUB), Commentary (COM), etc.
MetadataCluster	The group of all the metadata-only logical channels in an AVCluster.
MetadataControl	A logical object that is used to manipulate a specific metadata Property. Examples are Font Control, Color Control, etc.
MetadataControl Property	Parameter of a MetadataControl. Examples are Current, Next, Range Properties of a Font Control.
MetadataTrack	A group of MetadataChannels in a MetadataCluster that is associated with a single program.
MetadataTrack Selector	A Control that allows indicating or selecting one MetadataTrack from the MetadataCluster. The selected MetadataTrack is called the Active MetadataTrack.
Metadata Unit (DU)	Provides basic manipulation on the incoming logical MetadataChannels.
Mixer Unit (MU)	Mixes a number of logical input channels into a number of logical output channels.
MLP	Meridian Lossless Packing
MotionVector	A couple of coordinates (x,y) defining an offset from the current MacroBlock position in the current VideoFrame into the previous VideoFrame. The current MacroBlock position is defined by two coordinates (xMB, yMB) defining the position of the upper left pixel of that MacroBlock.
MPEG	Moving Pictures Expert Group.
MUD	Mixer Unit Description.
NAL	Network Abstraction Layer, See [IEC14496_10]
NAL Unit	An integer number of bytes, preceded by a ony-byte NAL Header indicating the type of data in the NAL Unit.
OTD	Output Terminal Description.
Output Pin	A logical output connection to an Entity. Carries a single AVCluster.
Output Terminal (OT)	An outlet for AV information flowing out of the AVCore.
Processing Unit (PU)	Applies a predefined process to a number of logical input channels.
PUD	Processing Unit Description.
Router Unit (RU)	Provides the means to assemble an outgoing AVCluster from multiple incoming AVClusters.
RUD	Router Unit Description.
Selector Unit (SU)	Selects from a number of input AVClusters.
ServiceInterval	A grouping of USB (micro)frames that are related.
ServiceIntervalPacket	A packet that contains all the VideoSamples that are transferred over the bus during a ServiceInterval.
SI	ServiceInterval.
SIP	ServiceIntervalPacket.
SIPDescriptor	A 4-byte field present at the beginning of every SIP, providing information about the layout of the SIP. Only present when transporting Extended Audio Format Type I data.
SOF	Start of Frame ([USB2.0]).
Stereo 3D Video	A video stream consisting of 2 separate video channels, providing a stereoscopic 3-dimensional video experience. One video channel is intended for the left eye, and the other video channel is intended for the right eye.
SubHeader	A header containing additional information, related to the data in the SIP.
SUD	Selector Unit Description.
Terminal	A logical object inside an AVCore that represents a connection to the AVCore's outside world.

Term	Description
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 (micro)frame.
Unit	A logical object inside an AVCore that represents a certain AV functionality.
VideoBundle	VideoChannels are physically organized into VideoTracks, and VideoTracks are then organized into the VideoBundle where each VideoVhannel has a VideoChannel Type associated with it. Very similar to the VideoCluster concept, but the physical characteristics of the video stream, such as VideoFrame Rate are retained in the VideoBundle.
(Logical) VideoChannel	A logical transport medium for a single video channel. Makes abstraction of the physical Properties and formats of the connection. Is identified by VideoChannel Type: i.e. observation location. Examples are Channels for left eye, right eye.
VideoChannel Type	The observation location of a VideoChannel. Uniquely identifies the VideoChannel. Examples are Channels for left eye (OL001), right eye (OL002).
VideoCluster	The group of all the video-only logical channels in an AVCluster.
VideoControl	A logical object that is used to manipulate a specific video Property. Examples are Contrast Control, Zoom Control, etc.
VideoControl Property	A parameter of a VideoControl. Examples are Current, Minimum, Maximum and Resolution Properties of a Brightness Control.
VideoFrame	One of the many still images which compose a complete moving picture or VideoSequence.
VideoParticle	The basic structure used to send video data over USB. Can be one VideoSample, two VideoSamples that share Chrominance information, or individual Luminance and Chrominance components of one VideoSample.
VideoPayload	All data bytes related to one VideoFrame. Includes NAL Headers, InfoBlock Headers, etc. The AVHeader is not part of the VideoPayload.
VideoSample	The basic representation of a video signal, sampled (digitized) at a specific moment in time.
VideoSequence	A sequence of closely related VideoFrames that together make up a movie or video clip.
VideoSlot	A collection of VideoSubSlots, each containing a VideoSample of a different physical video channel, taken at the same moment in time.
VideoStream	A concatenation of a potentially very large number of VideoSlots ordered according to ascending time where the VideoSamples are formatted according to one of the VideoFrame and VideoSample Formats described in this specification and where the video channels are organized in a VideoBundle.
VideoStreamConfig (VideoStream Configuration)	An XML Description that provides all the information necessary to fully characterize a VideoStream. Includes a VideoBundle, VideoFrame, and VideoSample component.
VideoStreamConfigList	A list of VideoStreamConfig Descriptions used to indicate the different VideoStream Configurations an AVData Entity supports.
VideoSubSlot	Holds a single VideoParticle.
Viewpoint	A position from which an event (program) is observed.
WMA	Windows Media Audio.
XML Descriptions	See Descriptions.

2. Fundamentals

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 ServiceInterval (see below) that would normally have an isochronous transfer. Both situations are considered equivalent and the AVFunction is expected to behave the same. In both cases, this specification considers a Transfer Delimiter to be an entity that can be signaled over the USB.

2.2. ServiceInterval and ServiceIntervalPacket Definitions

To better describe packetization for audio, the concept of a ServiceInterval (SI) is introduced. A ServiceInterval represents the number of USB (micro)frames or Bus Intervals within which an isochronous endpoint is serviced once.

For full-/high-speed isochronous endpoints, a ServiceInterval is defined as:

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

See the [USB2.0] specification for more information on the **bInterval** field, its allowed values and its use.

For SuperSpeed isochronous endpoints, a ServiceInterval is identical to the Service Interval as defined in the [USB3.0] specification:

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

In addition, a ServiceIntervalPacket (SIP) is introduced. A ServiceIntervalPacket is defined as a packet that contains all the samples that are transferred over the bus during a ServiceInterval. For full-/high-speed endpoints, the SIPs are exactly the same as the physical packets that are transferred over USB. For high-speed high-bandwidth endpoints, the SIP is the concatenation of the two or three physical packets that are transferred over the bus in a microframe. For SuperSpeed endpoints, the SIP is the concatenation of up to the 48 physical packets (3 burst opportunities of maximum 16 bursts) that are transferred over the bus in a ServiceInterval.

The above definitions provide a model of 'one packet per ServiceInterval', irrespective of the actual transactions on the USB.

2.3. Audio Transport

2.3.1. AudioSubSlot

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

This specification limits the possible AudioSubSlot sizes ($subSlotSize$) to 1, 2, 3 or 4 bytes per AudioSubSlot. An AudioSample is represented using a number of bits ($bitResolution$) less than or equal to the total number of bits available in the AudioSubSlot:

$$bitResolution \leq subSlotSize * 8$$

In case $bitResolution < subSlotSize * 8$, the actual AudioSample is always left-justified and padded with trailing zero bits to fill the entire AudioSubSlot.

AVData Audio Streaming endpoints shall be constructed in such a way that a valid transfer can take place as long as the reported AudioSubSlot size is respected during transmission. If the reported bits per sample ($bitResolution$) do not correspond with the number of significant bits actually used during transfer, the device will either discard trailing

significant bits ([actual_bits_per_AudioSample] > $bitResolution$) or interpret trailing zeros as significant bits ([actual_bits_per_AudioSample] < $bitResolution$).

2.3.2. AudioSlot

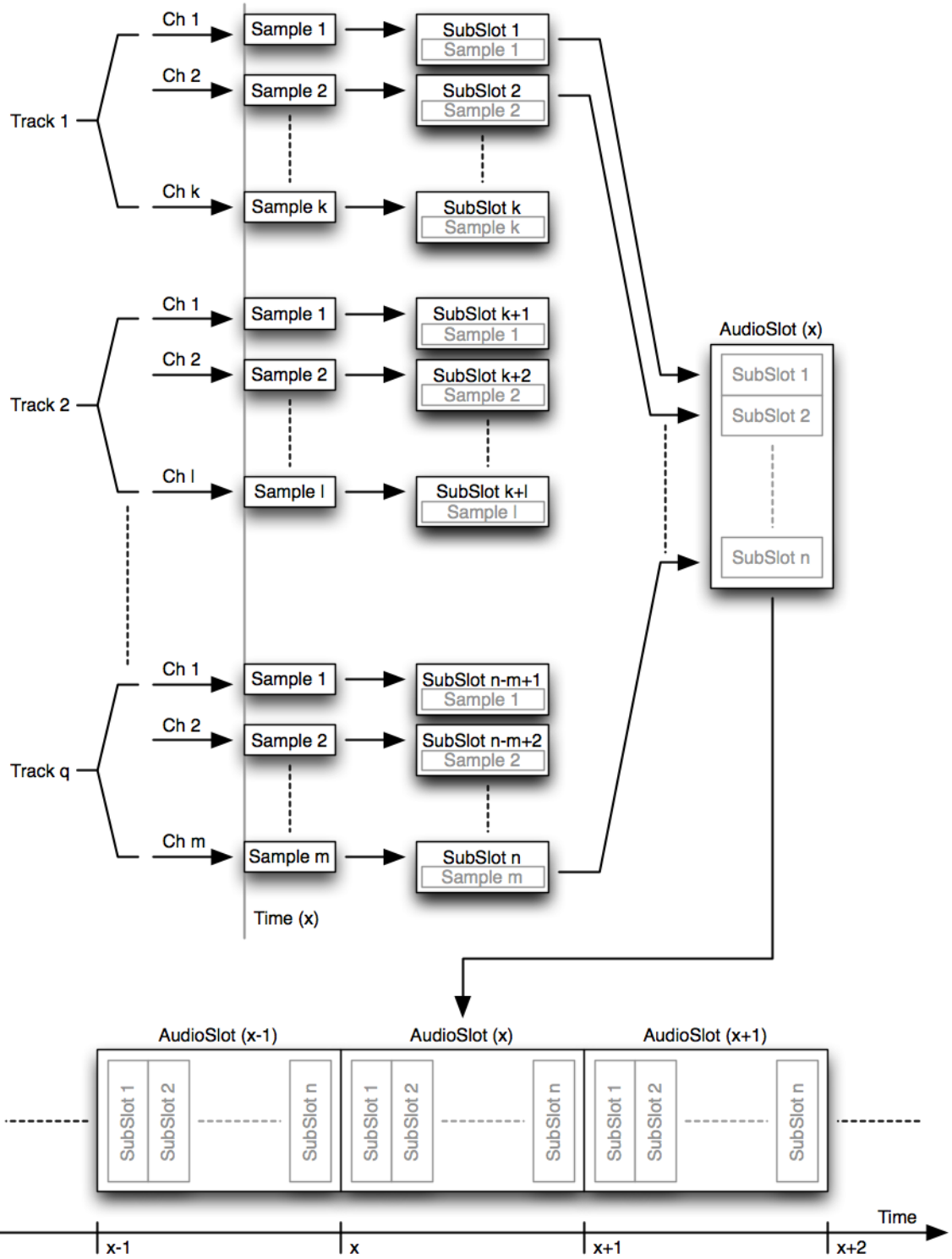

Figure 2-1: AudioStream

An AudioSlot consists of a collection of AudioSubSlots, each containing an AudioSample of a different physical audio channel, taken at the same moment in time, x. The number of AudioSubSlots in an AudioSlot equals the total number of audio channels in the AudioBundle. The ordering of the AudioSubSlots in the AudioSlot obeys the same rules as set forth in the USB AVFunction Definition for the logical channels and AudioTracks in an AudioCluster. The AudioSamples, taken at time x+1, are interleaved in the same fashion to generate the next AudioSlot and so on. The notion of physical channels is explicitly preserved during transmission. All AudioSubSlots shall have the same AudioSubSlot size.

2.3.3. AudioBundle

Audio channels are transmitted over USB in an AudioBundle. The AudioBundle concept is very similar to the AudioCluster concept as defined in the USB AVCore Definition, except that an AudioBundle retains all of the physical aspects of the AudioStream, such as sampling rate, bit resolution, etc. Like an AudioCluster, an AudioBundle consists of a number of AudioTracks and each AudioTrack in turn consists of a number of physical audio channels. The following figure illustrates the concept.

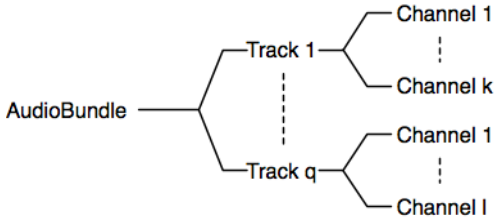

Figure 2-2: AudioBundle

An AudioBundle is a concatenation of a potentially very large number of AudioSlots, ordered according to ascending time. AudioBundles are packetized when transported over USB whereby SIPs can only contain an integer number of AudioSlots. 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 AudioSlots available to construct a SIP, a Transfer Delimiter shall be sent instead.

2.3.3.1. AudioBundle Description

The AudioBundle Description is a hierarchical component of the AudioStream Configuration Description. Refer to Section 2.9.1, “AudioStreamConfigList Description” for details.

2.4. AudioFrame

For the purposes of this specification, an AudioFrame is identical to an AudioSlot. An AudioFrame therefore consists of a collection of AudioSubSlots, each containing an AudioSample of a different physical audio channel, taken at the same moment in time and this for each available channel of each available AudioTrack in the AudioBundle.

2.4.1. AudioFrame Rate

Since the AudioFrame is identical to the AudioSlot, the AudioFrame Rate is identical to the Audio Sampling Frequency.

All audio-related frequency values shall tolerate at least ±1000 PPM inaccuracy to accommodate sampling clock inaccuracies.

2.4.2. AudioFrame Organization

This specification currently supports only one AudioFrame Organization as described in Section 2.3.2, “AudioSlot”.

2.4.3. AudioFrame Format

The AudioFrame Format is identical to the AudioSlot Format as described in Section 2.3.1, “AudioSubSlot” and Section 2.3.2, “AudioSlot”.

2.5. AudioSample Formats

There are currently two categories of AudioSample Format Types defined by this specification:

- Simple AudioSample Formats
- Extended AudioSample Formats

Within each of these categories, there are four Types defined

- Simple/Extended AudioSample Format Type I
- Simple/Extended AudioSample Format Type II (Deprecated)
- Simple/Extended AudioSample Format Type III
- Simple/Extended AudioSample Format Type IV (Deprecated)

A unique AudioSample format ID (AFID) identifies each AudioSample Format Type. In addition, a unique AudioSample Format Boot Mode Code (AFLVC) is defined for use while in Boot Mode. Allowed values for the AFID and the AFLVC can be found in Appendix A.1, "AudioSample Format Type I Identifiers" and Appendix A.2, "AudioSample Format Type III Identifiers".

2.6. Simple AudioSample Formats

2.6.1. Type I Formats

Type I formats deal with AudioStreams that are transmitted over USB and are constructed on an AudioSample-by-AudioSample basis. Each AudioSample is represented by a single independent symbol, contained in an AudioSubSlot.

This specification supports AudioSample bit resolutions ranging from 1 up to 32 bits per AudioSample. Each AudioSample is transported in an AudioSubSlot as described above.

The $bitResolution$ and $subSlotSize$ values shall always obey the following equation:

$$subSlotSize = \big((bitResolution - 1)\ DIV\ 8\big) + 1$$

2.6.1.1. Type I Format Type Description

The Format Type I Description is a hierarchical component of the AudioStream Configuration Description. Refer to Section 2.9.1, "AudioStreamConfigList Description" for details.

2.6.1.2. Type I Supported Formats

The following paragraphs list all currently supported Type I AudioSample Formats. The different Type I AudioSample Format Identifiers can be found in Appendix A.1, "AudioSample Format Type I Identifiers".

2.6.1.2.1. PCM Format

The PCM (Pulse Coded Modulation) format is the most commonly used AudioSample. 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 AudioSubSlot. The binary point is located to the right of the sign bit so that all values lie within the range [-1, +1).

2.6.1.2.2. IEEE_FLOAT Format

The IEEE_FLOAT format is floating point PCM and is based on the ANSI/IEEE-754 floating-point standard. Audio data is represented using the basic 32-bit wide single-precision format. For details, refer to [IEEE_754].

The data is conveyed over USB using 32 bits per sample ($bitResolution = 32$, $subSlotSize = 4$).

2.6.2. Type II Formats

Type II formats deal with those formats that do not preserve the notion of physical channels during the transmission over USB. Type II formats were defined in the Audio Device Class Definition Release 1.0 ([FORMATS1.0]) and further maintained in the Audio Device Class Definition Release 2.0 ([FORMATS2.0]). However, they are not included in this specification anymore and deprecated moving forward.

2.6.3. Type III Formats

Type III formats mix characteristics of Type I and Type II formats to transmit AudioStreams over USB. One or more non-PCM encoded AudioStreams are packed into "pseudo-stereo samples" and transmitted as if they were real stereo PCM AudioSamples. The sampling frequency of these pseudo samples matches the sampling frequency of the original PCM AudioStreams. Therefore, clock recovery at the receiving end is easier than it is in the case of Type II 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.

The IEC60958 standard specifies a widely used method of interconnecting digital audio equipment using a two-channel linear PCM audio interface. The IEC61937 standard describes a way in which the IEC60958 interface is 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 AudioStream. From a USB transfer standpoint, the data streaming over the interface looks exactly like two-channel 32 bit PCM audio data ($bitResolution = 32, subSlotSize = 4$).

2.6.3.1. Type III Format Type Description

The Format Type III Description is a hierarchical component of the AudioStream Configuration Description. Refer to Section 2.9.1, "AudioStreamConfigList Description" for details.

2.6.3.2. Type III Supported Formats

Refer to the ISO/IEC 60958 and ISO/IEC 61937 (several parts) specifications and other formats as listed in Appendix A.2, "AudioSample Format Type III Identifiers."

2.6.4. Type IV Formats

Type IV formats were defined in the Audio Device Class Definition Release 2.0 ([FORMATS2.0]). However, they are not included in this specification anymore and deprecated moving forward.

2.7. Extended AudioSample Formats

In addition to the Simple AudioSample Formats described above, Extended AudioSample Formats Type I and III are defined. These are based on the Simple AudioSample Formats Type I and III definitions but they provide an optional Header and for the Extended AudioSample 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 AudioSample 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:

Table 2-1: SIPDescriptor Layout

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

The **wFlags** 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 and shall be set to zero.

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 AudioStream consisting of a Control Stream only, without Header or audio data. It is also possible to create an AudioStream 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 sematics 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.

2.7.1. 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.

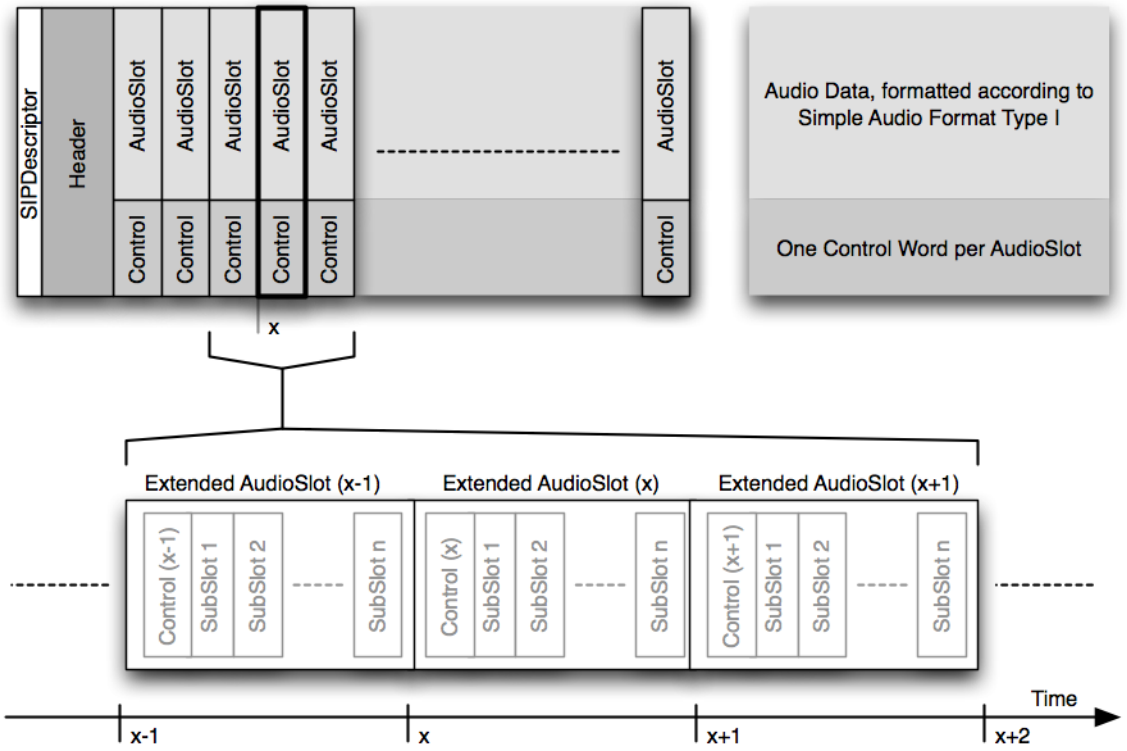

Figure 2-3: Extended Type I Format

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.

2.7.1.1. Extended Type I Format Type Description

The Extended Format Type I Description is a hierarchical component of the AudioStream Configuration Description. Refer to Section 2.9.1, "AudioStreamConfigList Description" for details.

2.7.2. Extended Type II Formats

Extended Type II formats are deprecated and not defined in this specification.

2.7.3. 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.

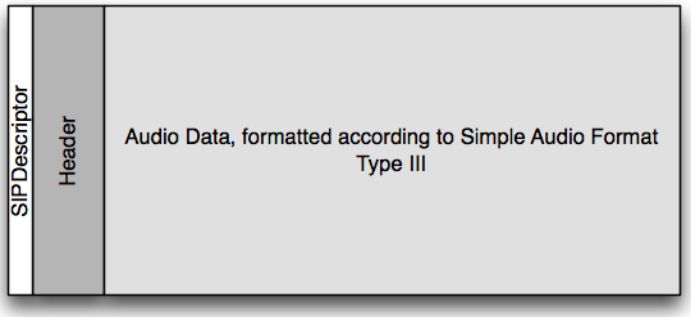

Figure 2-4: Extended Type III Format

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

2.7.3.1. Extended Type III Format Type Description

The Extended Format Type III Description is a hierarchical component of the AudioStream Configuration Description. Refer to Section 2.9.1, "AudioStreamConfigList Description" for details.

2.8. AudioStreamConfig

An AudioStream Configuration (AudioStreamConfig) is fully characterized by the following components:

- The AudioBundle
- The AudioFrame component, consisting of:
  - The AudioFrame Rate or Audio Sampling Frequency
- The AudioSample component, consisting of:
  - The AudioSample Format

The AudioBundle describes the number of AudioTracks in the AudioBundle, and for each AudioTrack, the number of audio channels in the AudioTrack. For each audio channel the Audio Channel Type (see [AVFUNCTION]) is indicated as well. See Section 2.3.3, "AudioBundle".

The AudioFrame Rate indicates the rate (frequency) at which consecutive AudioFrames (or AudioSlots) are produced or consumed. See Section 2.4.1, "AudioFrame Rate".

The AudioSample Format defines how the AudioSamples are represented and packetized into byte streams. See Section 2.6, "Simple AudioSample Formats" and Section 2.7, "Extended AudioSample Formats".

2.8.1. AudioStreamConfig Description

The AudioStreamConfig Description is a hierarchical component of the AudioStreamConfigList Description. Refer to Section 2.9.1, "AudioStreamConfigList Description" for details.

2.9. AudioStreamConfigList

An AudioStreamConfigList is a construct that consists of a list of AudioStreamConfig Descriptions that are supported by a particular AVData Audio Streaming interface in which the AudioStreamConfigList is advertised.

2.9.1. AudioStreamConfigList Description

The AudioStreamConfigList Description can be found at: AudioStreamConfigList.

3. Audio Synchronization

This specification describes how AudioStreams are transported over an isochronous pipe.

To keep the audio source and the audio sink synchronized in time, it is important that the average Sampling Frequency on both sides remain synchronized on average. Indeed, if the audio sink consumes the AudioSamples at an average rate that is higher than the rate at which the audio source provides the AudioSamples, then the audio sink will inevitably run out of audio data to render and buffer underruns will occur. Likewise, if the audio sink consumes the AudioSamples at an average rate that is lower than the rate at which the audio source provides the AudioSamples, then the audio sink will inevitably run out of buffer space to store the incoming audio data and buffer overruns will occur.

To avoid any of the scenarios described above, the *average* Sampling Frequency on both sides (audio source and sink) *must* match. Instantaneous Sampling Frequency may vary (drift) between audio source and sink and the accumulated variation is limited by the amount of buffer that is available at the audio sink.

3.1.1. Synchronization Types

Each isochronous audio endpoint used in an AVData Audio Streaming interface belongs to a synchronization type as defined in Section 5 of the USB Specification. The following sections briefly describe the possible synchronization types.

3.1.1.1. Asynchronous

Asynchronous isochronous audio endpoints produce or consume data at a rate that is locked either to a clock external to the USB or to a free-running internal clock. These endpoints cannot be synchronized to a start of frame (SOF), Bus Interval Boundary (BIB), or to any other clock in the USB domain.

3.1.1.2. Synchronous

The clock system of synchronous isochronous audio endpoints can be controlled externally through SOF or BIB synchronization. Such an endpoint shall lock its sample clock to the SOF or BIB tick.

3.1.1.3. Adaptive

Adaptive isochronous audio endpoints are able to source or sink data at any rate within their operating range. This implies that these endpoints run an internal process that allows them to match their natural data rate to the data rate that is imposed at their interface.

4. Audio Packetizing

Audio 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.

The goal should be to keep the instantaneous number of AudioSlots per SI, n_i as close as possible to the average number of AudioSlots 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 a SI and Δt is the sample time ($\frac{1}{F_s}$). 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 AudioSlot, that is, $\Delta n_i = 1$. This implies that all SIPs shall either contain $INT(n_{av})$ AudioSlots (small SIP) or $INT(n_{av}) + 1$ (large SIP) AudioSlots. 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 $T_{si} = 1ms$. Then $n_{av} = 44.1$ AudioSlots. Since the source can only send an integer number of AudioSlots per SI, it will send small SIPs of 44 AudioSlots. Each SI, it therefore sends '0.1 AudioSlot' too few and it will accumulate this fractional part in an accumulator. After having sent 9 small SIPs of 44 AudioSlots, at the tenth SI it will have exactly one AudioSlot in excess and therefore can send a large SIP containing 45 AudioSlots. 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 AudioSlots followed by 1 large SIP of 45 AudioSlots. The following table illustrates the process:

Table 4-1: 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 → 0.0
n+10	44.1	44	0.1	0.1
n+11	44.1	44	0.1	0.2
…	…	…	…	…

4.1. Audio Bursting

Although it is advisable for easy clock recovery to spread the AudioSamples as evenly as possible over time, sometimes there are more prevalent reasons, such as link power management considerations, to group the AudioSamples in bursts so that the links on the USB between the Host and the AVFunction can go into low power mode in between bursts. However, bursting the audio data inherently increases latency since a larger number of AudioSamples needs to be collected before the burst can be sent over the USB. Also, there is a negative impact on the amount of buffer memory that needs to be present to enable bursting. Furthermore, the reserved bandwidth on the USB for this endpoint is higher than what is needed in the non-bursting case.

The Controller can determine what level of bursting to use based on several parameters:

- The level of bursting-induced latency the application that is using the isochronous pipe can tolerate. For example, for simple music-only playback a much larger audio latency can be tolerated than is the case for playback of a movie soundtrack.
- The **wMaxPacketSize** value for the isochronous endpoint.
- The amount of available memory to store AudioSamples during one SI. For SuperSpeed, this is the **wBytesPerInterval** value. For full- and high-speed endpoints, the AVDD contains an `<audioSIPSize>` element that is equivalent to the SuperSpeed **wBytesPerInterval** field.
- The **bInterval** value for the isochronous endpoint.
- For isochronous OUT transfers, the Controller can regulate the burst period by inserting 'no data' Transfer Delimiters. For Isochronous IN transfers, the Controller has to periodically (at the specified $2^{(bInterval-1)}$ rate) poll the isochronous pipe, thereby somewhat limiting the effective burst period.

If an AVFunction desires to enable some form of audio bursting, it shall expose at least two Alternate Settings (besides the default Alternate Setting 0); one optimized for low latency that does not uses audio bursting and one optimized for low power and therefore higher latencies, using audio bursting. This way the Controller can choose the appropriate Alternate Setting depending on the latency requirements of the application that is currently using the isochronous pipe. Also, the mandatory presence of a non-bursting Alternate Setting guarantees interoperability with a Controller that does not support audio bursting.

4.2. Pitch Control

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

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

Pitch control is restricted to adaptive endpoints only. AVData Audio Streaming interfaces that support pitch control on their isochronous endpoint are required to report this in the Clock Domain Description (as part of the AVData Audio Streaming Interface Description) by indicating that they support the Pitch Control to enable or disable the pitch control functionality.

5. Auxiliary Protocols

5.1. HDCP Protocol

The HDCP protocol uses the HDCP SubHeader to convey the periodic Link Synchronization information, required [HDCP2.1] (see Section 2.6, Link Synchronization for details). [HDCP2.1] 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 5-1: HDCP SubHeader Layout

Offset	Field	Size	Value	Description
0	**bLength**	1	Number	Size of this SubHeader, in bytes.
1	**bSubHeaderID**	1	Number	HDCP_ENCRYPTION
2	**bOffset**	1	Number	Offset to first full16-byte encrypted data block in the SIP.
3	**bReserved**	1	Number	Reserved for future extensions. Shall be set to zero.
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.

Each AudioStream Configuration (see Section 2.8, "AudioStreamConfig") reports in its AudioStreamConfig Description whether the HDCP Protocol is supported for that AudioStream Configuration through the `<@hdcp>` attribute.

5.2. Timestamp Protocol

The Timestamp (TS) protocol uses the Timestamp SubHeader to convey high-resolution time information over the isochronous pipe. The occurrence rate of the TS SubHeader is implementation-dependent.

The layout of the TS SubHeader is as follows:

Table 5-2: Hi-Res Timestamp Layout

Offset	Field	Size	Value	Description
0	**bLength**	1	Number	Size of this SubHeader, in bytes.
1	**bSubHeaderID**	1	Number	TIMESTAMP.
2	**bmFlags**	2	Bitmap	D0: Valid. D31..1: Reserved. Shall be set to 0.
4	**dReserved**	4	Number	Reserved. Shall be set to 0.
4	**qNanoSeconds**	8	Number	Offset in nanoseconds from the beginning of the AudioStream.

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

The **bSubHeaderID** field contains the TIMESTAMP constant to uniquely identify this SubHeader as a TS SubHeader.

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 **qNanoSeconds** field of the TS SubHeader shall be ignored.

The **dReserved** field is reserved for future extensions and shall be set to zero.

The **qNanoSeconds** field indicates the time T at which the first sample in the SIP needs to be rendered with respect to the start of the stream (T = 0). The **qNanoSeconds** field can range from 0 to (2^{64}-1) ns (unsigned 64-bit number). It is up to the entity that generates the timestamp to decide to which accuracy the timestamp will be rendered.

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 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 shall be taken to ensure that the information contained in the TS SubHeader is at all times in agreement with the implicit timing information, delivered by the isochronous streaming mechanism.

Each AudioStream Configuration (see Section 2.8, “AudioStreamConfig”) reports in its AudioStreamConfig Description whether the Timestamp Protocol is supported for that AudioStream Configuration through the `<@timestamp>` attribute.

Appendix A. Additional AV Device Class Codes

A.1. AudioSample Format Type I Identifiers

Table A-1: Format Type I Identifiers

Name	Identifier Value (AFID)	AFLVC
PCM	PCM	0x0001
IEEE_FLOAT	IEEE_FLOAT	0x0002

A.2. AudioSample Format Type III Identifiers

Table A-2: Format Type III Identifiers

Name	Identifier Value (AFID)	AFLVC
IEC61937_AC-3	AC-3	0x0003
IEC61937_MPEG-1_Layer1	MPEG-1_LAYER1	0x0004
IEC61937_MPEG-1_Layer2/3 or IEC61937_MPEG-2_NOEXT	MPEG-1_LAYER2-3 or MPEG-2_NOEXT	0x0005
IEC61937_MPEG-2_EXT	MPEG-2_EXT	0x0006
IEC61937_MPEG-2_AAC_ADTS	MPEG-2_AAC_ADTS	0x0007
IEC61937_MPEG-2_Layer1_LS	MPEG-2_LAYER1_LS	0x0008
IEC61937_MPEG-2_Layer2/3_LS	MPEG-2_LAYER2-3_LS	0x0009
IEC61937_DTS-I	DTS-I	0x000A
IEC61937_DTS-II	DTS-II	0x000B
IEC61937_DTS-III	DTS-III	0x000C
IEC61937_DTS-IV	DTS-IV	0x000D
IEC61937_ATRAC	ATRAC	0x000E
IEC61937_ATRAC2/3	ATRAC2-3	0x000F
IEC61937_WMA	WMA	0x0010
IEC61937_WMA_PRO	WMA_PRO	0x0011
IEC61937_E-AC-3	E-AC-3	0x0012
IEC61937_MAT_MLP	MAT_MLP	0x0013
IEC61937_MPEG-4_AAC_LATM	MPEG-4_AAC	0x0014
IEC61937_MPEG-4_HE_AAC_LATM	MPEG-4_HE_AAC	0x0015
IEC61937_MPEG-4_HE_AAC_V2_LATM	MPEG-4_HE_AAC_V2	0x0016

A.3. SubHeader Codes

Table A-3: SubHeader Codes

SubHeader Code	Value
HEADER_UNDEFINED	0x00
HDCP_ENCRYPTION	0x01
TIMESTAMP	0x02

A.4. AV Format 2 General Constants

Table A-4: AV Format 2 General Constants

Constant Identifier	Value
HDCP_PACKET_HEADER_TIME	512 (ms)