UNIVERSAL SERIAL BUS
DEVICE CLASS DEFINITION
FOR
AUDIO DEVICES

Release 4.0

April 2023

SCOPE OF THIS RELEASE

This document is the Release 4.0 of this Device Class Definition.

CONTRIBUTORS

Chris March	Apple Inc.
Rhoads Hollowell	Apple Inc.
Girault Jones	Apple Inc.
Geert Knapen (Co-Chair & Editor)	Apple Inc. E-mail: gknapen@apple.com
Eric Schulz	Apple Inc.
Abdul Rahman Ismail (Co-Chair)	Intel Corporation
Devon Worrell	Intel Corporation
Niel Warren	Google
Terry Moore	MCCI Corporation
Egidio Sburlino	Microsoft Corporation
Franz Detro	Native Instruments
Mike Kent	Roland
Brad Saunders	SpecWerkz
Bob Dunstan	SpecWerkz
Morten Christiansen	Synopsys
Paul E. Berg (Program Manager)	USB-IF

REVISION HISTORY

Rev.	Date	Filename	Description
1.0	Mar. 1998	Audio10.pdf	Release 1.0
2.0	May 2006	Audio20 final.pdf	Release 2.0
3.0	Sep. 2016	Audio30.pdf	Release 3.0
4.0	Apr. 2023	Audio40.pdf	Release 4.0

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

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1 INTRODUCTION

The following sections provide a brief overview of the scope and purpose of this document. It also includes a list of related documents.

1.1 SCOPE

The Audio Device Class Definition applies to all Devices or Functions embedded in composite Devices that are used to manipulate audio, voice, and sound-related functionality. This includes both audio data (analog and digital) and the functionality that is used to directly manipulate audio signals, such as Gain and Tone Control. The Audio Device Class does not include functionality to operate transport mechanisms that are related to the reproduction of audio data, such as tape transport mechanisms or CD-ROM drive control, nor does it include how HID features (volume up/down, play/pause, etc.) are conveyed. See *Universal Serial Bus Device Class Definition for Human Interface Devices (HID)* for more related information.

1.2 PURPOSE

The purpose of this document is to describe the minimum capabilities and characteristics an Audio Function shall support to comply with the USB Audio Device Class. This document also provides recommendations for optional features.

1.3 RELATED DOCUMENTS

- USB Core Specifications:
  - *Universal Serial Bus 2.0 Specification*, Revision 2.0 (referred to in this document as the *USB 2 Specification*). See Chapter 5, “USB Data Flow Model” and Chapter 9, “USB Device Framework.”
  - *Universal Serial Bus 3.2 Specification*, Revision 1.0 (referred to in this document as the *USB 3 Specification*). This document covers details specific to SuperSpeed and SuperSpeed+ Devices.
  - *Universal Serial Bus 4.0 Specification,* Version 2.0. This document covers details specific to Gen T Devices.
- *Universal Serial Bus Device Class Definition for Human Interface Devices (HID),* Version 1.11.
- 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)
- Windows Media Audio (WMA) specification. (available from http://www.microsoft.com)
- Digital Audio Compression Standard (AC-3), ATSC A/52A Aug. 20, 2001. (available from http://www.atsc.org/)
- ANSI/IEEE-754 standard. *IEEE Standard for Floating-point Arithmetic*.
- ISO/IEC 60958 International Standard: *Digital Audio Interface and Annexes.*
- ISO/IEC 61937 standard. *Interface for non-linear PCM Encoded audio bitstreams applying IEC 60958*
- ISO/IEC 80000-13 standard. *Quantities and Units*.
- ITU G.711 standard. *Pulse code modulation (PCM) of voice frequencies*.
- ETSI Specification TS 102 114, “DTS Coherent Acoustics; Core and Extensions”. (Available from
- EN 50332-2:2013 specification. Available from several sources. (https://www.en-standard.eu/)
- EN 50332-2:2017 specification. Available from several sources. (https://www.en-standard.eu/)
- IETF RFC 4122 GUID definition: Available from https://www.ietf.org/rfc/rfc4122.txt
- HDCP 2.3 HDCP Specifications: Available from https://www.digital-cp.com/hdcp-specifications.
- IEEE Std 269-2019. "IEEE Standard for Measuring Electroacoustic Performance of Communication Devices”.

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 the *USB Core Specifications*.

AC-3	Audio compression standard from Dolby Labs.
AEC	Acronym for Audio Echo Cancellation.
Armed AudioContol	An AudioControl that has its NEXT Attribute preloaded with a valid value.
ASRC	Acronym for Asynchronous Sampling Rate Converter.
Audio data stream	Transport medium that carries audio information.
Audio Function	Independent part of a USB Device that deals with audio-related functionality.
Audio Function Descriptor Set (AFDS)	The collection of the Interface Association Descriptor and the full set of Standard Descriptors and class-specific Descriptors that together comprise the entire Audio Function.
Audio Interface Association (AIA)	Grouping of a single AudioControl Interface along with zero or more AudioStreaming Interfaces that together constitute a complete interface to a particular version (compliance level) of the Audio Function.
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 continuous sequence of advancing, time-ordered audio slots. Typically, a set of one or more channels that have a common rate, direction, and phase.
Audio Subslot	Holds a single PCM audio sample.
AudioControl	Logical object that is used to manipulate a specific audio property. Examples are Gain Control, Mute Control, etc.
AudioControl Attribute	Parameter of an AudioControl. Examples are Current, Next, Minimum, Maximum and Resolution Attributes of a Gain Control.
AudioControl Interface	USB interface used to access the AudioControls inside an Audio Function.
AudioStreaming Interface	USB interface used to control the USB transport of audio streams into or out of the Audio Function.
Bus Interval	Time interval between two consecutive SOF tokens while the Bus is in L0. This is not the same as the Service Interval.
byte	An 8-bit quantity. Byte fields start with the letter b (or ba for arrays), followed by the field name.

ChannelGroup	A grouping of audio channels inside of a Cluster that carry tightly related audio.
Clock Domain	A zone within the Audio Function that is served by sampling clocks that are all derived from the same Reference Clock.
Clock Selector (CX)	Selects from several input clock signals.
Clock Source (CS)	Generates an audio-related clock signal (Main clock or sample clock).
Cluster	A grouping of audio channels that carry tightly related audio, streaming over a connection inside the Audio Function. An instantiation of a Cluster Configuration.
Active Cluster	A grouping of audio channels that currently carries valid audio.
Inactive Cluster	A grouping of audio channels that currently does not carry valid audio.
Incoming Cluster	The Cluster entering an Entity on a single Input Pin. Note that some Entities have multiple Input Pins and therefore have multiple incoming Clusters.
Outgoing Cluster	The Cluster leaving an Entity on its single Output Pin.
Cluster Configuration	The definition of the Cluster as detailed in the Cluster Descriptor.
Cluster Descriptor	An Extended Descriptor describing the characteristics of a Cluster, such as the number of channels, Purpose, and Relationship of each channel.
Connector Entity	Provides access to the AudioControl(s) that impact the behavior of the Connector.
Device	Term used to describe a device that has USB functionality.
DTS	Acronym for Digital Theater Systems.
DVD	Acronym for Digital Versatile Disc.
dword	A 4-byte quantity. Dword fields start with the letter d (or da for arrays), followed by the field name.
Effect Unit (EU)	Provides advanced audio manipulation on the incoming logical audio channels.
Encoded Audio Bit Stream	A continuous sequence of advancing, time-ordered encoded audio frames.
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.
Entity	Addressable logical object inside an Audio Function.

EntityGroup	A logical grouping of Entities that are associated and have a relationship with one another.
Extended Descriptor	A new representation for class-specific Descriptors that allows for potentially large (>255 bytes) and dynamic Descriptors.
Extension Unit (XU)	Applies an undefined process to a set of logical input channels.
Feature Unit (FU)	Provides basic audio manipulation on the incoming logical audio channels.
IAD	Acronym for Interface Association Descriptor.
ICN	Acronym for Input Channel Number. The ordinal number of an audio channel in a Cluster on an Input Pin of an Entity.
Input Pin	Logical input connection to an Entity. Carries a single Cluster.
Input Terminal (IT)	Receptacle for audio information flowing into the Audio Function.
Interface Association	A group of two or more Interfaces associated with the same Audio Function.
IPN	Acronym for Input Pin Number. The ordinal number of an Input Pin of an Entity.
kbits	kilobits. Equals 1000 bits per ISO/IEC 80000-13 standard.
KiB	kibibyte. Equals 1024 bytes per ISO/IEC 80000-13 standard.
Logical Audio Channel	Logical transport medium for a single audio channel. Makes abstraction of the physical properties and formats of the connection. Is usually identified by spatial location. Examples are Front Left channel, Surround Array Right channel, etc.
LSb	Least Significant bit.
Mixer Unit (MU)	Mixes one or more logical input channels into one or more logical output channels.
MPEG	Acronym for Moving Pictures Expert Group.
MSb	Most Significant bit.
Node	A location within an Entity's internal topology, identified by an OCN:ICN:IPN Triplet.
OCN	Acronym for Output Channel Number. The ordinal number of an audio channel in a Cluster on the Output Pin of an Entity.
OCN:ICN:IPN Triplet	The Output Channel Number, Input Channel Number, and Input Pin Number combination that identifies a specific AudioControl within an Entity's internal topology.
Output Pin	Logical output connection from an Entity. Carries a single Cluster.
Output Terminal (OT)	An outlet for audio information flowing out of the Audio Function.

PCM	Acronym for Pulse Coded Modulation.
Pin Channel	A physical channel a Unit or Terminal supports/implements on an Input or Output Pin.
Pin Channel Count (PCC)	The maximum number of physical channels a Unit or Terminal supports/implements on an Input or Output Pin.
Phase Locked Loop	A hardware or software control system that generates an output signal whose phase is related to the phase of an input signal. Typically used in clock recovery applications.
Power Domain	A grouping of one or more Entities whose power states are simultaneously controlled by a single Power Domain Entity.
Power Domain Entity	Provides access to the AudioControl(s) that impact the behavior of the Power Domain.
Processing Unit (PU)	Applies a predefined process to one or more logical input channels.
Reference Clock	The single clock generator that is used to derive all clocks in a Clock Domain.
Reserved	**“**Reserved” is a keyword indicating reserved bits, bytes, words, fields, and code values that are set-aside for future standardization. Their use and interpretation may be specified by future extensions to this specification and, unless otherwise stated, shall not be utilized, or adapted by vendor implementation. A reserved bit, byte, word, or field shall be set to zero by the sender and shall be ignored by the receiver.
Request Error	The Device returning a STALL PID when an error is detected in the content of the Request. See Section 9.2.7, “Request Error”, of the Universal Serial Bus Specification Revision 2.0 for more details.
Sampling Rate Converter Unit (RU)	Converts the incoming audio data stream, running at a sampling rate that is synchronous to a first Main clock, into an outgoing audio data stream that is running at a sampling rate that is synchronous to a second Main clock, which is either free running with respect to the first Main clock (asynchronous sampling rate conversion) or synchronous to the first Main clock (synchronous sampling rate conversion).
Selector Unit (SU)	Selects from several input Clusters.
Service Interval	The period between consecutive transfers on a periodic Endpoint, specified by the value in the **bInterval** field of the Endpoint Descriptor. This is not the same as the Bus Interval.
Service Interval Packet	A packet that contains all the audio slots that are transferred over the bus during a Service Interval.
Terminal	Addressable logical object inside an Audio Function that represents a connection to the Audio Function’s outside world.

Unit	Addressable logical object inside an Audio Function that represents a certain audio sub-functionality.
USB-C®	A 24-pin USB connector system.
WMA	Acronym for Windows Media Audio.
word	A 2-byte quantity. Word fields start with the letter w (or wa for arrays), followed by the field name.

2 MANAGEMENT OVERVIEW

The USB is very well suited for transport of audio, ranging from low fidelity voice connections to high quality, multi-channel audio streams. The USB has become a ubiquitous connector on modern computers and is well-understood by most consumers today. As such, it has become the connector of choice for many peripherals and is indeed the simplest and most pervasive digital audio connector available today. Consumers can count on this medium to meet all of their current and future audio needs. Many applications from communications, to entertainment, to music recording and playback, can take advantage of the audio features of the USB.

In principle, a versatile bus specification like the USB provides many ways to propagate and/or control audio signals. For the industry, however, it is very important that audio transport mechanisms be well-defined and standardized on the USB. Only in this way can interoperability be guaranteed among the many possible Audio Devices on the USB. Standardized audio transport mechanisms also help keep software drivers as generic as possible. The Audio Device Class described in this document satisfies those requirements. It is written and revised by experts in the audio field. Other Device Classes that address audio in some way should refer to this document for their audio interface specification.

An essential issue in audio is synchronization of the data streams. Indeed, the smallest artifacts are easily detected by the human ear. Therefore, a robust synchronization scheme on isochronous transfers has been developed and incorporated in the *USB Core Specifications*. The Audio Device Class definition adheres to these synchronization schemes to transport audio data reliably over the bus.

This document contains all necessary information for a designer to build a USB-compliant Device that incorporates Audio functionality. It specifies the standard and class-specific Descriptors that shall be present in each USB Audio Function. It further explains the use of class-specific Commands that allow for full Audio Function control. Several predefined data formats are listed and fully documented. Each format defines a standard way of transporting audio over the USB.

Many of the changes introduced in this version of the USB Specification for Audio Devices are inspired by the desire to use USB Audio in modern portable devices. Special attention has been paid to make the Audio Device Class more power-friendly by providing new tools to selectively enable and disable parts of the Audio Function and by supporting burst mode data transfers for longer sleep times in between data transfers. In addition, the specification supports new CODEC types and data formats for consumer audio applications, provides numerous clarifications of the original specification and extensions to support various changes in the core specification.

Inherent restrictions that were present in previous versions have been alleviated. For example, it was impossible to have a Cluster with more than 61 channels on the Input Pin of a Feature Unit due to the limited descriptor space (< 256 bytes) available to describe per-channel AudioControls.

This ADC 4.0 specification supersedes the ADC 3.0 version. It is highly recommended to design devices using this latest 4.0 version as the previous version imposed some restrictions (especially in the realm of backwards compatibility) that made it cumbersome to implement. The use of the ADC 3.0 version is therefore highly discouraged.

2.1 OVERVIEW OF KEY DIFFERENCES BETWEEN ADC 2.0 AND ADC 4.0

The following list is not an exhaustive list of all changes that have been introduced. For complete information, refer to the full specification.

- Dolby Processing Unit removed
- Encoder and Decoder support removed

- Copy protection (S/PDIF-style) removed
- bmAttributes (MaxPacketsOnly bit) field in class-specific isochronous Endpoint Descriptor removed
- Type II and Extended Type II and Type IV Audio Data Formats removed
- Additional Type III formats introduced
- New Power Domains
- New Multi-Function Processing Unit
- New Cluster Descriptor
- New Terminal Companion Descriptor (previously named Extended Terminal Descriptor in ADC 3.0)
- New Extended Descriptors (previously named High Capability Descriptors in ADC 3.0)
- Introduced a new backwards compatibility mechanism
- New CAP and NEXT Attribute for AudioControls
- New Commit Control
- New ChannelGroups and EntityGroups
- Modified the meaning of the bm(a)Controls field and renamed to bm(a)OptControls
- Selector Units now support an off position
- Several Descriptors have a slightly different layout and some added fields
- Active and Inactive Clusters
- Clarified difference between Cluster channels and Pin channels
- AudioStreaming Interfaces now exclusively used for USB Streaming Interfaces
- Removed bAssocTerminal from Terminal Descriptor. Replaced with EntityGroup concept
- Removed Terminal Types
- Converted most class-specific Descriptors from Traditional Descriptors to Extended Descriptors
- Added Channel ID to enable channel traceability throughout the Audio Function
- EN 50332-2 and EN 50332-3 support
- Added Channel Remap PU
- Optional support for LPM/L1 and clarified the concept
- Added StartDelay concept to AudioStreaming Interfaces
- Removed LockDelay concept from endpoint descriptors. Replaced by the StartDelay concept
- Eliminated class-specific Endpoint Descriptors
- Removed Overrun, Underrun, Pitch Controls
- Additional sources for interrupts
- New class-specific String Descriptors
- Added Connector Entities
- Removed the RAW_DATA concept
- New Control Command structure to overcome the limitations in allowed parameter bit widths in the current Control request structure
- Removed the MEM Attribute and its associated infrastructure
- Redefined the Clock Source Entity
- Removed the Input Gain Control
- Renamed the Volume Control to Gain Control

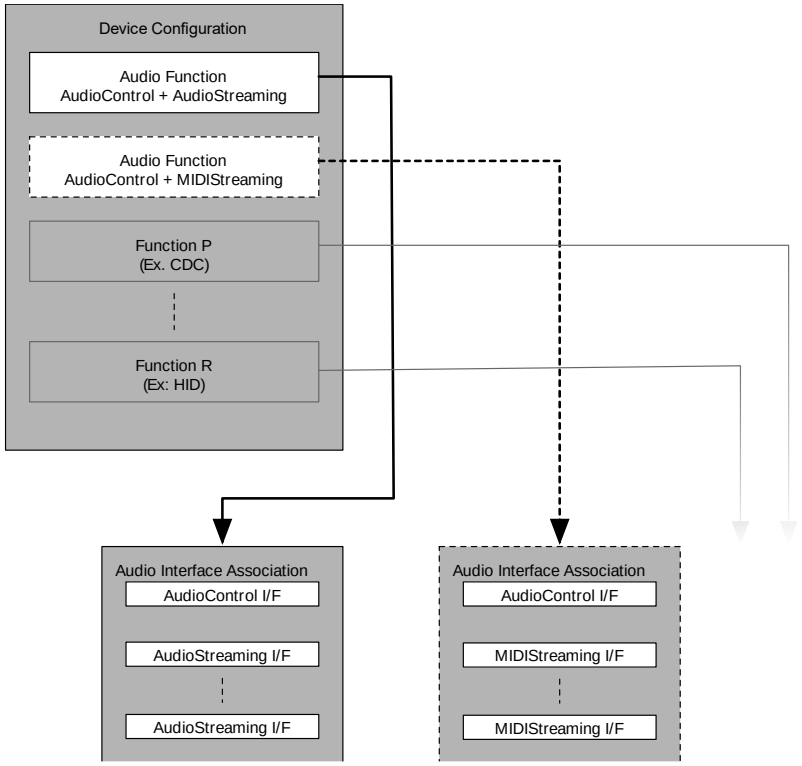
3 FUNCTIONAL CHARACTERISTICS

The following sections describe the functional characteristics of a USB Audio Function.

3.1 INTRODUCTION

In many cases, Audio functionality is incorporated with other USB Class functionality within a single (composite) Device. The Audio Function is thus located at the interface level in the Device Class hierarchy. The following figure provides details.

Figure 3-1: Audio Interface Associations and Interfaces

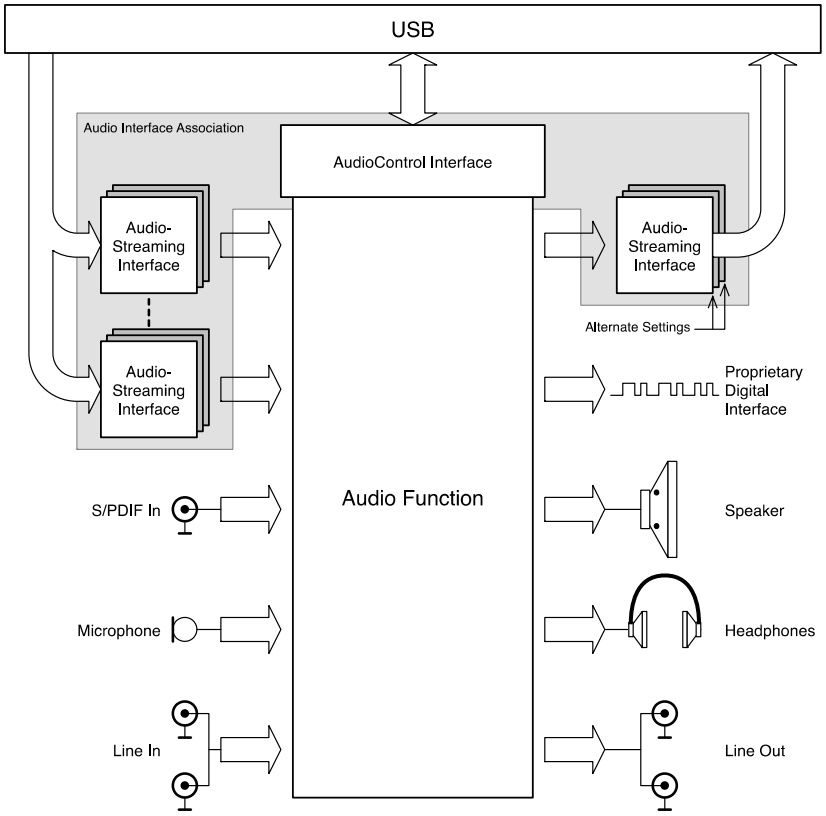


An Audio Function is considered to be a ‘closed box’ that has very distinct and well-defined interfaces to the outside world. Audio Functions are described through an Audio Interface Association (AIA). The AIA groups all USB interfaces that together provide access to the Audio Function for control and streaming purposes.

An AIA shall have a single AudioControl Interface and may have zero or more AudioStreaming Interfaces. The AudioControl (AC) Interface is used to access the AudioControls inside the Audio Function whereas the AudioStreaming (AS) Interfaces are used to encapsulate their associated USB Endpoint(s) and provide access to AudioControls that impact the transport of USB audio streams into and out of the Audio Function.

The following figure illustrates the concept of an Audio Function and its associated interfaces:

Figure 3-2: Example Audio Function Global View



A Device may have *multiple independent* Audio Functions located in the same composite Device. They are each accessed through their own Audio Interface Association.

All functionality pertaining to controlling parameters that directly influence audio perception (like volume) are located inside the central rectangle and are exclusively controlled through the AudioControl Interface. Streaming aspects of the communication to or from the Audio Function are handled through separate AudioStreaming Interfaces, as necessary. Each USB audio stream shall be represented by an AudioStreaming Interface. All control information that is related specifically to the streaming behavior of the USB interface is conveyed through the AudioStreaming Interface.

Also note that the connection between the AudioStreaming Interfaces and the Audio Function is not 'solid'. The reason for this is that when seen from the inside of the Audio Function, each audio stream entering or leaving the Audio Function is represented by a special object, called a Terminal (see further). The Terminal concept abstracts the actual AudioStreaming Interface inside the Audio Function and provides a logical view on the connection rather than a physical view. This abstraction allows audio channels within the Audio Function to be treated as 'logical' audio channels that do not have physical characteristics associated with them anymore (analog vs. digital, format, sampling rate, bit resolution, etc.).

Non-USB audio streams, such as an S/PDIF connection, shall never be represented by an AudioStreaming Interface.

3.2 BASIC AUDIO DEVICE EXAMPLES

The Basic Audio Device Definition documents that were introduced in ADC 3.0 have been deprecated and are replaced with example documents that outline for each of the previous Basic Audio Device Definition Device Types

how they should be implemented using the concepts defined in this ADC 4.0 specification. The appropriate Descriptor Sets for each are included and should provide implementers with a ready-to-use implementer's guide.

3.3 BACKWARDS COMPATIBILITY

This 4.0 version of the *Audio Device Class Definition* (ADC 4.0) as such is not backwards compatible with any of the previous versions of the same specification. However, a method to provide the broadest interoperability with existing ecosystems has been incorporated into this specification and is described in detail in Section 8, "Backwards Compatibility Considerations."

3.4 AUDIO INTERFACE ASSOCIATION (AIA) AND INTERFACE ASSOCIATION DESCRIPTOR

An Audio Function is completely defined by its interfaces. The standard USB Interface Association mechanism is used to describe the Audio Interface Association, i.e. binding those interfaces together. An Audio Function shall have exactly one Audio Interface Association (AIA). Each AIA shall have one AudioControl Interface and may have zero or more AudioStreaming Interfaces. All Audio Function interfaces shall be part of an Audio Interface Association, even if the Audio Interface Association contains only a single AudioControl Interface.

Interface Association is expressed via the standard USB Interface Association Descriptor (IAD). Every Interface Association Descriptor has a **bFunctionClass**, **bFunctionSubClass** and **bFunctionProtocol** field that together identify the function that is represented by the Association. The following paragraphs define these fields for the Audio Device Class.

3.4.1 AUDIO FUNCTION CLASS

The Audio Function Class is contained in the **bFunctionClass** field of a standard Interface Association Descriptor. This specification requires that the Function Class code be the same as the Audio Interface Class code.

The Audio Interface Class code and therefore the Audio Function Class code is assigned by the USB-IF. The assigned codes can be found in Appendix A.2, "Audio Function Class Code."

3.4.2 AUDIO FUNCTION SUBCLASS

The Audio Function Subclass is contained in the **bFunctionSubClass** field of a standard Interface Association Descriptor. This field is not used by this specification and shall be set to FUNCTION_SUBCLASS_UNDEFINED.

The assigned code can be found in Appendix A.3, "Audio Function Subclass Codes." of this specification. All other Subclass codes are unused and reserved by this specification for future use.

3.4.3 AUDIO FUNCTION PROTOCOL

The Audio Function Protocol is contained in the **bFunctionProtocol** field of a standard Interface Association Descriptor. The Function Protocol code is used to reflect the compliance level of the Audio Function so that enumeration software can decide which driver version needs to be instantiated. This specification requires that the Function Protocol code be AF_VERSION_04_00.

The assigned Protocol codes can be found in Appendix A.4, "Audio Function Protocol Codes" of this specification. All other Protocol codes are unused and reserved by this specification for future use.

3.5 AUDIO INTERFACE CLASS

The Audio Interface Class code is contained in the **bInterfaceClass** field of the standard interface Descriptor of any Interface in an AIA. All USB Interfaces which provide functionality through this specification use the same Audio Interface Class code.

The Audio Interface class code is assigned by the USB. The assigned code can be found in Appendix A.5, “Audio Interface Class Code”.

3.6 AUDIO INTERFACE SUBCLASS

The Audio Interface Subclass code is contained in the **bInterfaceSubClass** field of the standard interface Descriptor of any Interface in an AIA. All USB interfaces that provide functionality through this specification shall use one of the following Audio Interface Subclass codes as provided by this specification:

- AUDIOCONTROL Interface Subclass
- AUDIOSTREAMING Interface Subclass

The assigned codes can be found in Appendix A.6, “Audio Interface Subclass Codes” of this specification. All other Subclass codes are unused and reserved by this specification for future use.

3.7 AUDIO INTERFACE PROTOCOL

The Audio Interface Protocol code is contained in the **bInterfaceProtocol** field of the standard interface Descriptor of any Interface in an AIA. The Interface Protocol code is used to reflect the compliance level of the Audio Function.

All Audio Functions compliant with this ADC 4.0 specification shall use Interface Protocol Code IP_VERSION_04_00. The assigned codes can be found in Appendix A.7, “Audio Interface Protocol Codes” of this specification. All other Protocol codes are unused and reserved by this specification for future use.

3.8 CLOCK DOMAINS

A Clock Domain is defined as a zone within which all sampling clocks are derived from the same Reference clock. Therefore, within the same Clock Domain, all sampling clocks are frequency-locked, and their timing relationship is constant. However, the sampling clocks may be at different sampling frequencies, but related exactly by the ratio of two integers. The Reference Clock may be generated in many ways. An internal crystal could be used as the Reference Clock; the USB start of frame (SOF) or even an externally supplied clock could serve as the Reference Clock. Different sampling clocks within the same Clock Domain may then be derived from the Reference Clock through a combined use of Clock Source and Clock Selector Entities.

Multiple different Clock Domains may exist within the same Audio Function. Clock Domains are described via a topology map, as part of the various Clock Source Descriptors that are members of the Clock Domain.

3.9 POWER DOMAINS

A Power Domain is defined as a zone within the Audio Function that groups one or more Entities and allows the Host to control power consumption levels for that Domain. This way, the Host can potentially switch parts of the Audio Function to lower power states when these parts are currently not in use, leading to an overall decrease in power consumption. As an example, a USB headset may have two separate Power Domains, one for the output-related functionality (headphones) and one for the input-related functionality (microphones). When the headset is used to simply listen to music, the input-related functionality may be temporarily switched to a low power state

(or even completely off) to conserve power. This is especially important if the headset is used in conjunction with a battery-powered device, such as a mobile phone, where extending battery life is important.

Multiple different Power Domains may exist within the same Audio Function. Power Domains are identified by a Power Domain Entity with a unique Power Domain ID within the Audio Function.

3.10 GROUPS

Three types of Groups are defined by this specification:

- ChannelGroup: Part of a Cluster. A subset of channels in the Cluster that are logically related. A ChannelGroup shall have only channels as its Members.
- EntityGroup: A collection of *Entities* that the Audio Function wishes to advertise as functionally or physically related. An EntityGroup shall have only Entities as its Members.
- CommitGroup: A collection of AudioControls, potentially from multiple Units, that the Audio Function marks for synchronous updates, using the Commit Capability.

Group nesting (of any type) is prohibited: i.e., a Group shall never be a Member of another Group.

Multiple different Groups may exist within the same Audio Function.

3.11 AUDIO SYNCHRONIZATION TYPES

Each isochronous audio Endpoint used in an AudioStreaming Interface belongs to a synchronization type as defined in Section 5 of the *USB Core Specification*. The following sections briefly describe the possible synchronization types.

3.11.1 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 are not synchronized to a start of frame (SOF).

3.11.2 SYNCHRONOUS

The clock system of synchronous isochronous audio Endpoints can be controlled externally through SOF or Bus Interval synchronization. Such an Endpoint shall lock its sample clock to the SOF tick marking the start of a new Bus Interval.

3.11.3 ADAPTIVE

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

3.11.4 IMPLICATIONS OF THE DIFFERENT SYNCHRONIZATION TYPES

The following sections provide some information on the impact the different synchronization types have on both the Device and Host implementation. Several scenarios are considered and for each synchronization type, the implications on Host and Device side are listed.

3.11.4.1 SINGLE DIRECTION SINK ENDPOINT

A typical example for this scenario is a USB speaker, implementing a sink Endpoint that receives streaming audio data from the Host.

- Asynchronous
  - **Host**: The Host driver needs to be able to handle an explicit feedback Endpoint. From the feedback data, the Host then decides how many samples to send over the data streaming Endpoint in subsequent Service Intervals.

    Note: The expected value returned by the feedback Endpoint is the number of full and partial samples expected on average for every Service Interval expressed by the **bInterval** field of the Isochronous data Endpoint. See the description of feedback Endpoints in the appropriate USB core specification for more information about the format of feedback Endpoints.

  - **Device**: The Device has its own local, free-running audio sample clock, which determines how many samples are consumed by the Device each Service Interval. The Device shall implement an explicit feedback Endpoint as well as the necessary logic to provide the correct feedback values to send over said Endpoint back to the Host. The advantage of this mode of operation is that it is rather easy to generate a robust, stable, low-jitter, high-quality audio sample clock (derived from a crystal-based Main clock, for example).

    Note: The expected value returned by the feedback Endpoint is the number of full and partial samples expected on average for every Service Interval expressed by the **bInterval** field of the Isochronous data Endpoint. See the description of feedback Endpoints in the appropriate USB core specification for more information about the format of feedback Endpoints.

- Synchronous
  - **Host**: The Host needs to send out a known number of bytes for each packet going to the Device. The Host may need to generate a (fixed) pattern of audio samples to achieve the desired sampling rate. As an example, to generate a sampling rate of 44.1 kHz in a Full-Speed implementation, the Host needs to send a repeating pattern of nine packets containing 44 audio samples, followed by one packet containing 45 audio samples.
  - **Device**: This synchronization type requires the Device to implement either an audio clock PLL or an ASRC.
- Adaptive
  - **Host**: The Host may use any method or means to determine how many samples per Service Interval to transmit. Effectively operating as a “Synchronous-to-SOF” Source is an easy approach, but not the only one allowed by the USB core specification.
  - **Device**: This synchronization type requires the Device to implement either an audio clock PLL or an ASRC to adapt to the average number of samples arriving over a certain period.

3.11.4.2 SINGLE DIRECTION SOURCE ENDPOINT

A typical example for this scenario is a USB microphone, implementing a source Endpoint that sends streaming audio data to the Host.

- Asynchronous
    - **Host**: The Host needs to operate as an Adaptive Sink. Depending on the Host's system design, this may require an ASRC or an audio clock PLL on the Host side.
    - **Device:** The Device has its own local, free-running audio sample clock, which determines how many samples are produced by the Device each Service Interval. The advantage of this mode of operation is that it is rather easy to generate a robust, stable, low-jitter, high-quality audio sample clock (derived from a crystal-based Main clock, for example).
- Synchronous
    - **Host**: The Host receives a known number of audio samples in each packet coming from the Device. The Host should expect a fixed pattern of audio samples to achieve the desired sampling rate.
    - **Device**: This synchronization type requires the Device to implement either an audio clock PLL to lock on to the SOF or the start of Bus Interval and generate a high-quality audio sample clock directly or use an ASRC as a bridge between the local and USB Clock Domains. The Device shall generate a fixed packet size pattern as described in Section 7, "Audio Data Formats."
- Adaptive (not recommended for implementation)
    - **Host:** This scenario requires the implementation of a feedforward OUT Endpoint in the Device that allows the Host to inform the Device how many samples per Service Interval to send to the Host over the streaming data IN Endpoint.

        Note: Declaring a source endpoint as Adaptive without providing the feedforward OUT endpoint is a violation of the USB core specification.
    - **Device:** This synchronization type requires the Device to implement an audio clock PLL or an ASRC to adapt to the sample rate being communicated by the Host via the feedforward OUT Endpoint.

3.11.4.3 DUAL DIRECTION SOURCE AND SINK ENDPOINT PAIR

A typical example for this scenario is a USB headset, implementing both a sink Endpoint that receives streaming audio data from the Host and a source Endpoint that sends streaming audio data to the Host.

In general, the Endpoints may be treated independently, using the same rules and guidelines as stated above. This includes the ability for the Device to support separate clocks on each Endpoint.

However, if the Device has its own clock and both data streams share this clock, the following two special cases can be used to allow implicit feedback:

- Both Endpoints are Asynchronous. In this case, the data rate that appears on the Source Endpoint can be used by the Host as implicit feedback to adjust the data rate transmitted to the Sink Endpoint.
- Both Endpoints are Adaptive. In this case, the data rate that appears on the Sink Endpoint can be used by the Device to adjust the data rate transmitted by the Source Endpoint.

Note: Using either of these two implicit feedback mechanisms would preclude the ability of an ADC 4.0 compliant Device from being able to use Power Domains to shut down either the source or the sink, as data shall remain flowing in both directions to provide the feedback/feedforward information.

3.12 INTER CHANNEL SYNCHRONIZATION

An important issue when dealing with audio, and 3-D audio in particular, is the phase relationship between different physical audio channels. Indeed, the virtual spatial position of an audio source is directly related to and influenced by the phase differences that are applied to the different physical audio channels used to reproduce the audio source. Therefore, it is imperative that USB Audio Functions respect the phase relationship among all related audio channels. However, the responsibility for maintaining the phase relation is shared among the USB host software, hardware, and all the audio peripheral Devices or functions.

3.13 AUDIO FUNCTION TOPOLOGY

To be able to manipulate the physical properties of an Audio Function, its functionality is divided into addressable Entities where each Entity shall have a unique non-zero Entity ID. The AudioControl Interface encapsulates these Entities and is itself addressed via its interface number and Entity ID zero. AudioStreaming Interfaces are also addressed through their interface number and Entity ID zero.

This specification defines several different Entity types:

- Units
- Terminals
- Clock Entities
- Connector Entities
- Power Domain Entities

Units provide the basic building blocks to fully describe the internals of most Audio Functions. Audio Functions are built by connecting several of these Units. A Unit has one or more Input Pins and a single Output Pin, where each Pin carries a Cluster of logical audio channels (see Section 3.13.2, “Cluster”).

Units are wired together by connecting their I/O Pins according to the required topology. Note that it is perfectly legal to connect the Output Pin of an Entity to multiple Input Pins residing on different other Entities, effectively creating a one-to-many connection.

In addition to Units, Terminals are defined as well. There are two types of Terminals. The Input Terminal (IT) is an Entity that represents a starting point for audio channels inside the Audio Function. The Output Terminal (OT) represents an ending point for audio channels inside the Audio Function. From the Audio Function’s perspective, a USB streaming Endpoint is a typical example of an audio source or sink that is represented by an Input or Output Terminal. It either provides USB data streams to the Audio Function (IT) or consumes data streams coming from the Audio Function (OT). Likewise, a Digital to Analog converter, built into the Audio Function is represented as an Output Terminal in the Audio Function’s model.

The Input Terminal connects into the Audio Function through the Input Terminal’s single Output Pin. The Audio Function connects to the Output Terminal through the Output Terminal’s single Input Pin.

Input Pins of a Unit are numbered starting from one up to the total number of Input Pins on the Unit. The single Output Pin number is always one.

Input Pin one is somewhat special in that it is considered the ‘dominant’ Input Pin on most Units. This means that for both single input and multi-input Units that have a Unit Bypass function, the Cluster that enters the Unit on Input Pin one shall be passed unaltered to the Output Pin whenever the Unit Bypass function is engaged. It is always possible to override this standard behavior by not implementing the optional Bypass Control and instead

provide a bypassing Selector Unit. When an Entity has both a Cluster Control and a Bypass Control, the Bypass Control takes precedence when engaged.

Input Terminals have only one Output Pin and its number is always one. Output Terminals have only one Input Pin and it is always numbered one.

The Clusters traveling over I/O Pins and their interconnects are not necessarily of a digital nature. It is perfectly possible to use the Unit model to describe fully analog or even hybrid Audio Functions. The mere fact that I/O Pins are connected is a guarantee (by construction) that the protocol and format, used over these connections (analog or digital), is compatible on both ends.

Also, depending on certain Audio Function settings, such as some parts of the Audio Function going to a low power state, or a Selector Unit's Selector Control set to the no-connect (n.c.) position, or the absence of valid clock signal on a Terminal, etc., it is possible that an Entity's Output Pin does not carry a valid audio signal, i.e. the Cluster on that Output Pin is Inactive (see Section 3.13.2, "Cluster"). In that case, any Entity that receives this Inactive Cluster on one of its Input Pins shall disregard any signaling on that Input Pin and internally drive Silence on *all* its physical channels it is designed to support on that Input Pin (irrespective of the advertised number of logical channels in the Inactive Cluster).

Clusters may change dynamically over time. For example, a Selector Unit that has two Input Pins where Input Pin one receives a two-channel Cluster and Input Pin two receives a six-channel Cluster, then the Selector Unit's Output Pin will switch between a two-channel an a six-channel Cluster whenever the Unit's Selector Control is flipped between position one and two.

An Audio Function is allowed to autonomously switch the Cluster on any of its internal connections from one of the supported Cluster Configurations to another supported Cluster Configuration on that node. However, if an Output Terminal that is associated with an AudioStreaming Interface experiences a change in Cluster Configuration at its Input Pin or that Cluster becomes Inactive, then that AudioStreaming Interface shall switch to Alternate Setting zero.

The effect on the behavior of a physical output interface when a Cluster Configuration change occurs at the Input Pin of the Output Terminal that represents that interface is implementation dependent.

Units and Terminals by design have a maximum number of channels they support on any of their I/O Pins. This number is called the Pin Channel Count or PCC of an I/O Pin. The channels themselves are called Pin channels to distinguish them from the logical channels in the Cluster that travels over the Pin.

For all connections within the Audio Function, the PCC of the Output Pin of any Unit or Terminal shall match the PCC of every Input Pin to which that Output Pin is connected. Rather than advertising the (identical) PCC for each individual Pin that participates in a connection, the PCC is advertised in an Entity's Descriptor as a parameter associated with its Output Pin (the single source of the connection).

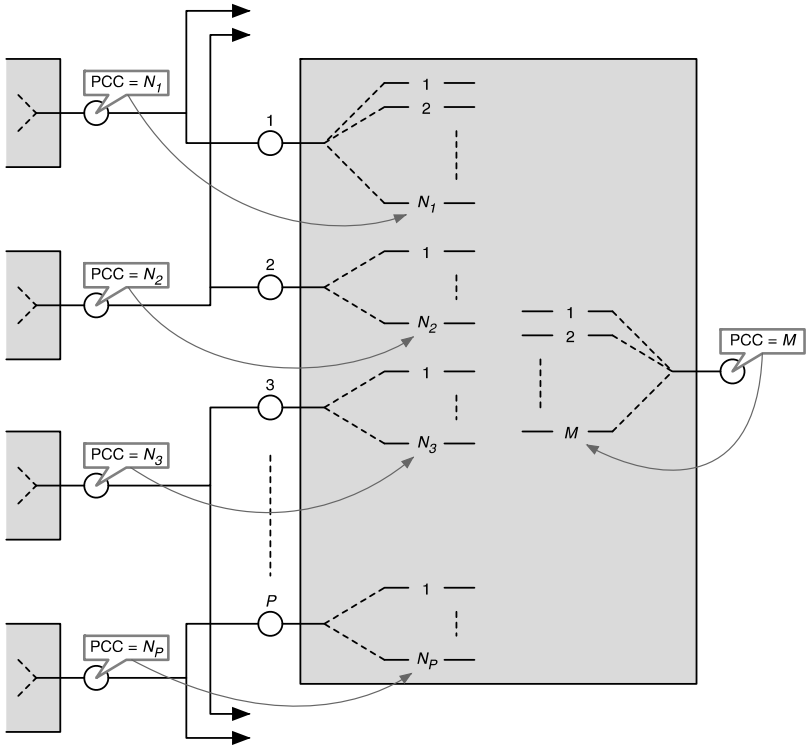
At any given time, the actual Cluster flowing over a connection between an Output Pin and the connected Input Pin(s) shall never have more logical channels than indicated by the PCC value.

Note: It is allowed for implementations to advertise a PCC value that is larger than strictly needed to support each of the available Clusters on a connection, i.e., the advertised PCC value may be larger than the maximum of the number of channels in each of the available Clusters.

Internally, Units and Terminals inherit the PCC on each of their Input Pins from the Entity's Output Pin to which the Input Pin is connected, and the number of Pin channels on each Input Pin is determined by that number. The

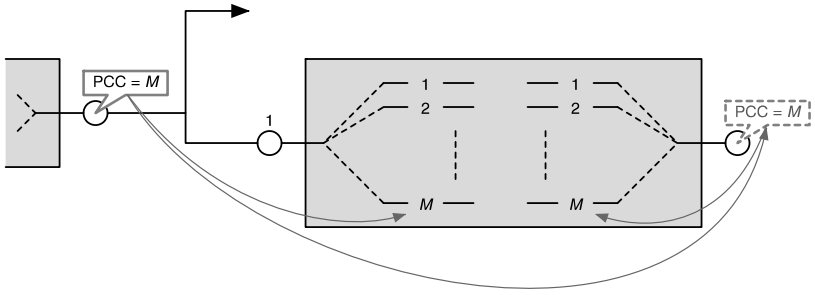
number of Output Pin channels is determined directly by the PCC value advertised for that Output Pin. The following figure illustrates this.

Figure 3-3: General PCC Inheritance Rules



For single-input Units that do not alter the number of Pin channels between their Input and Output Pin, the Unit does not advertise the Output Pin PCC value. Rather, the PCC value is inherited from upstream and determined by the first Entity upstream that defines a PCC value on its Output Pin as illustrated below.

Figure 3-4: Single Input Pin PCC Inheritance Rules



Note that in some cases multiple single-input Units may need to be traversed before finding the upstream Entity that defines a PCC value on its Output Pin.

Any signaling on currently unused Pin channels on an Input Pin shall be disregarded and the Unit or Output Terminal shall internally use Silence on those unused Pin channels for any processing within the Unit or Output Terminal.

Likewise, If the Unit or Terminal receives an Inactive Cluster on an Input Pin, or if the clock on an Output Terminal's Clock Input Pin is currently invalid, it shall disregard any signaling on that Input Pin and internally drive Silence on *all* of its Input Pin channels.

Every Unit in the Audio Function is fully described by its associated Unit Descriptor. The Unit Descriptor contains all necessary fields to identify and describe the Unit. Likewise, there is a Terminal Descriptor for every Terminal in the Audio Function. In addition, these Descriptors provide all necessary information about the topology of the Audio Function. They fully describe how Terminals and Units are interconnected.

This specification describes the following types of standard Units and Terminals that are considered adequate to represent most Audio Functions:

- Input Terminal (IT)
- Output Terminal (OT)
- Mixer Unit (MU)
- Selector Unit (SU)
- Feature Unit (FU)
- Sampling Rate Converter Unit (RU)
- Effect Unit (EU)
- Processing Unit (PU)
- Extension Unit (XU)

Besides Units and Terminals, the concept of a Clock Entity is introduced. Three types of Clock Entities are defined by this specification:

- Clock Source Entity (CS)
- Clock Selector Entity (CX)

A Clock Source Entity provides a certain sampling clock frequency to all or part of the Audio Function. A Clock Source Entity may represent an internal sampling frequency generator, but it may also represent an external sampling clock signal input to the Audio Function. An internal sampling frequency generator can be either asynchronous or it can be derived from the USB SOF signal. In addition, it can use a Clock Domain Reference Clock that may be shared with other Clock Source Entities within the Audio Function. This would indicate to the Host that these Clock Source Entities, although they potentially produce sampling clocks at different sampling frequencies, are frequency-locked.

A Clock Source Entity has a single Clock Output Pin that carries the sampling clock signal, represented by the Clock Source Entity. The Clock Output Pin number is always one.

A Clock Selector is used to select between multiple sampling clock signals that may be available inside an Audio Function. It has multiple Clock Input Pins and a single Clock Output pin. Clock Input Pins are numbered starting from one up to the total number of Clock Input Pins on the Clock Selector. The Clock Output Pin number is always one.

By using a combination of Clock Source and Clock Selector Entities, complex clock systems can be represented and exposed to Host software.

Clock Input and Output Pins are fundamentally different from Input and Output Pins defined for Units and Terminals. Clock Pins carry only clock signals and therefore cannot be connected to Unit or Terminal Input and Output Pins. They are only used to express clock circuitry topology.

Each Input and Output Terminal has a single Clock Input Pin that may be connected to a Clock Output Pin of a Clock Entity. The clock signal carried by that Clock Output Pin determines at which sampling frequency the hardware represented by the Terminal is operating. If there is no need to expose to the Host which clock signal is used by a Terminal or if the Terminal represents an analog port in the system, then the Clock Input Pin of the Terminal may be left unconnected.

A Sampling Rate Converter Unit has two Clock Input Pins that are typically connected to the Clock Output Pins of two different Clock Entities. The clock signals carried by those Clock Output Pins determine the sampling frequencies between which the Sampling Rate Converter Unit is converting and whether the conversion is synchronous (the two clock signals are derived from the same Main clock) or asynchronous (the two clock signals are derived from different, independent, Main clocks).

Each Clock Entity is described by a Clock Entity Descriptor. The Clock Entity Descriptors contain all necessary fields to identify and describe the Clock Entities. In addition, these Descriptors provide all necessary information about the clock topologies within the Audio Function.

A physical Connector, typically user-accessible, is represented by a Connector Entity and described by a Connector Entity Descriptor. The Connector Entity is always associated with one or more Input and/or Output Terminals through which the signals, present on the Connector, enter or leave the Audio Function.

Connector Entities and Power Domain Entities are slightly different in concept from the other Entities in the sense that they do not convey topological information. They are conceptual constructs that provide addressability for the AudioControls that reside within them, and they are described by a Connector Entity Descriptor or Power Domain Entity Descriptor respectively.

The Descriptors are further detailed in Section 4, “Descriptors” of this document.

The ensemble of AudioControl Interface Descriptor, AudioStreaming Interface Descriptors, Endpoint Descriptors, Unit Descriptors, Terminal Descriptors, Clock Entity Descriptors, Connector Entity Descriptors, and Power Domain Entity Descriptors provide a full description of the Audio Function to the Host. This information is typically retrieved from the Device at enumeration time. By parsing the Descriptors, an Audio Class driver should be able to fully control the Audio Function.

Important Note:

> The complete set of Audio Function Descriptors provides only a static initial description of the Audio Function. During operation, several events may happen that force the Audio Function to change its state. Host software shall be notified of these changes to remain ‘in sync’ with the Audio Function at all times. An extensive interrupt mechanism is in place to report all state changes to Host software.

This specification defines a set of symbols to graphically represent the building blocks, discussed above. This allows the creation of standardized topology diagrams to describe Audio functionality. See Figure 3-5, “Inside the Audio Function” for an example diagram. Input Terminals are typically located at the far left, while Output Terminals are at the far right of the topology diagram. The symbols, representing Audio Function Entities have their Input Pins on the left side of the symbol, and their single Output Pin on the right side. Input Pins are always numbered in ascending order, starting with Input Pin 1 in the upper left corner of the symbol, so that there is no need to explicitly provide Input Pin number labels on the symbols.

Figure 3-5, “Inside the Audio Function” illustrates some of the concepts defined above. Using the symbols defined further, it describes a hypothetical Audio Function that incorporates 15 Entities: three Input Terminals, five Units, three Output Terminals, three Clock Source Entities, and a Clock Selector Entity. Each Entity has its unique ID (from 1 to 15) and Descriptor that fully describes the functionality of the Entity and how that particular Entity is

connected into the topology of the Audio Function. Note that the AudioControl Interface itself is also considered an implicit addressable Entity with unique ID zero.

Input Terminal 1 (IT 1) is the representation of a USB OUT Endpoint used to stream audio from the Host to the Audio Device. IT 2 is the representation of an analog Line-In connector on the Audio Device whereas IT 3 is an analog Microphone-In connector on the Audio Device. Selector Unit 4 (SU 4) selects between the audio coming from the Host and the audio present at the Line-In connector. Feature Unit 5 (FU 5) is then used to manipulate the audio (Gain, Bass, Treble …) before it is presented to Output Terminal 9 (OT 9). OT 9 is the representation of a Headphone Out jack on the Audio Device.

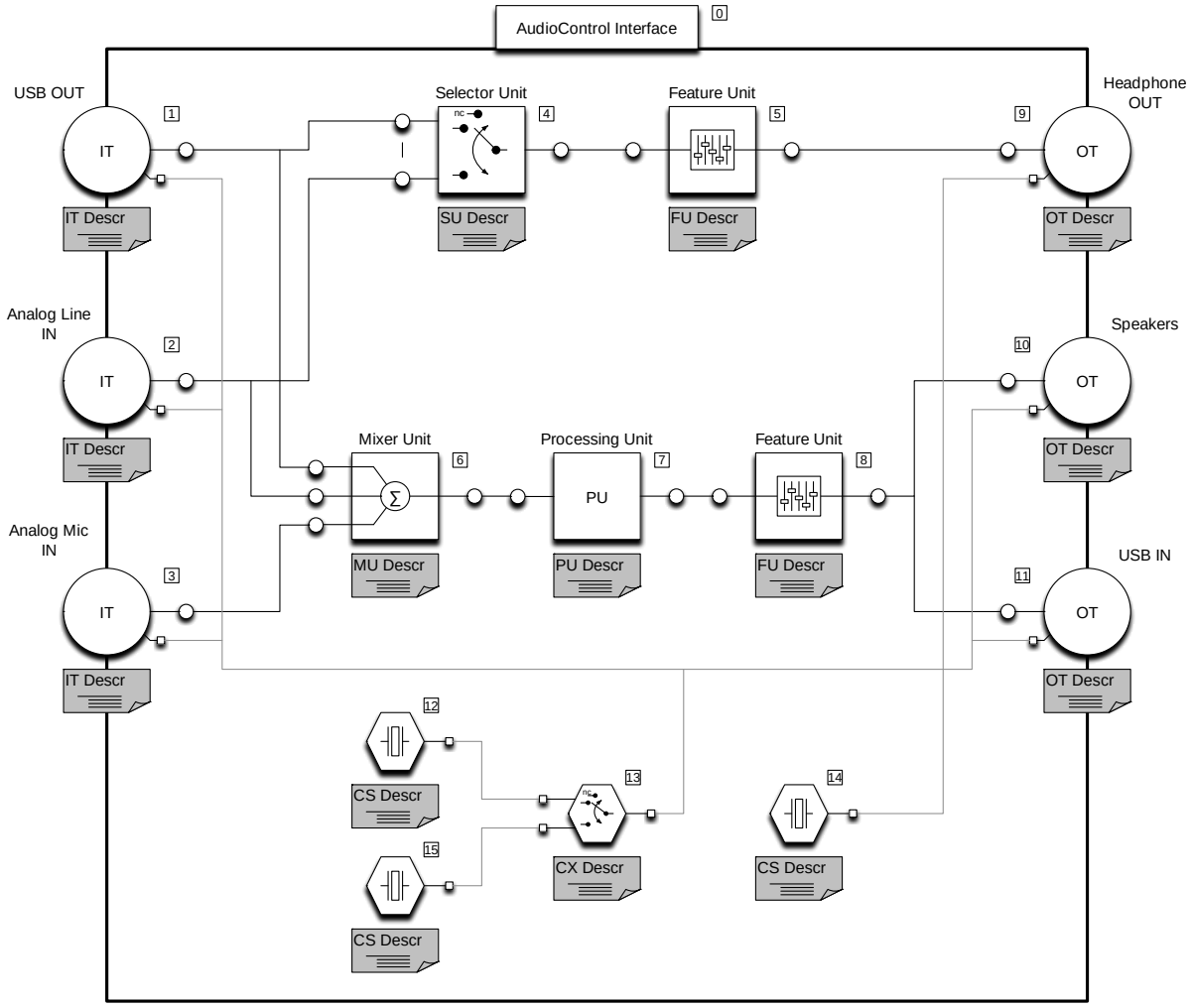
At the same time, all three input sources (USB OUT, Line-In, and Microphone-In) are connected to a Mixer Unit 6 (MU 6) that effectively mixes the three sources together. The output of the Mixer is then fed into a Processing Unit 7 (PU 7) that performs some audio processing algorithm(s) on the mix. The result is in turn sent to FU 8 where some final adjustments to the audio (Gain …) are made. FU 8 is connected to OT 10 and OT 11. OT 10 represents speakers incorporated into the Audio Device and OT 11 represents a USB IN Endpoint used to send the processed audio to the Host for recording purposes.

Clock Source Entity 12 (CS 12) represents an internal sampling frequency generator, running at 96 kHz for instance. Clock Source Entity 15 (CS 15) is the representation of an external reference sampling clock input that may be used to synchronize the Device to an external source. Clock Selector Entity 13 (CS 13) enables selection between the two available Clock Source Entities. The output of CS 13 provides a sampling frequency to IT 1, IT 2, IT3, OT 10, and OT 11 of 96 kHz. Clock Source Entity CS 14 further provides a sampling frequency of 48 kHz to OT 9 for driving the headphone. Since all sampling frequencies used inside the Audio Function are at all times derived from a single Main clock (internal or external) as indicated in the Clock Source Entity Descriptors, all audio streams in the Audio Function are synchronous.

The Descriptors, associated with each Entity clearly indicate to the Host what the exact nature of each Entity is. For instance, the IT 2 Descriptor contains a field that indicates to the Host that it represents an external connector on the Device, used as an analog Line-In. Likewise, the MU 6 Descriptor has a field that indicates that its Input Pin 1 is connected to the Output Pin of IT 1, Input Pin 2 is connected to the Output Pin of IT 2, and Input Pin 3 is connected to the Output Pin of IT 3.

For further details on Descriptor contents, refer to Section 4, “Descriptors” of this document.

Figure 3-5: Inside the Audio Function



3.13.1 AUDIOCONTROLS

Inside an Entity, functionality is further described through AudioControls. An AudioControl typically provides access to a specific audio or clock property. Each AudioControl has a set of Attributes that can be manipulated or that present additional information about the behavior of the AudioControl. An AudioControl has the following Attributes:

- Capability (CAP) Attribute (Mandatory)
- Current (CUR) Attribute (Mandatory)
- Next (NEXT) Attribute (Optional)

For details about AudioControl Attributes and their Read-Write privileges, see Section 5.3.2.2, “AudioControl Attributes” and Section 5.3.2.3, “AudioControl Read/Write Privileges.”

As an example, consider a Gain Control inside a Feature Unit. By issuing the appropriate Pull Commands, the Host software can obtain values for the Gain Control’s Attributes and, for instance, use them to correctly display the Gain Control to the user. All relevant information that Host software needs to interact with the Gain Control can be retrieved via a Pull Command to the CAP Attribute. The Gain Control’s CUR Attribute allows the Host software to

directly change the level setting of the Control. Setting the Control's NEXT Attribute allows the Host software to prepare a change to the setting of the Control. This change will only take effect when Host software issues a Commit Command. For more details, refer to Section 3.14.3, "AudioControls."

3.13.2 CLUSTER

A Cluster is a grouping of audio channels that carry tightly related synchronous audio information and that travels over the connections among Terminals and Units. Inside the Audio Function, complete abstraction is made of the actual physical representation of the audio in the Cluster. Each audio channel in the Cluster is a logical channel and all the physical attributes of the channel (sampling frequency, bit width, bit resolution, etc.) are not specified and considered irrelevant in the context of the Audio Function as seen through the AudioControl Interface. The fact that an Input Pin and an Output Pin are connected in the Audio Function's topology is a guarantee (by construction) that the audio flowing out of the Output Pin on the source Entity is compatible with and consumable by the Input Pin of the receiving Entity or Entities. This may involve some conversion processes that happen "behind the scenes". The details of these conversion processes, however, are beyond the scope of the logical view exposed by the Audio Function.

Channel numbering in the Cluster starts with channel one up to the number of channels in the Cluster. Channel Number zero is used to reference the Primary channel. See Section 3.14.3, "AudioControls" for more details.

A typical Cluster is characterized by several Attributes:

- A unique identifier for the Cluster
- The number of audio channels in the Cluster
- For each Channel in the Cluster:
  - The purpose of the audio channel, such as Voice, Speech, etc.
  - The audio channel relationship (or spatial location)
  - A Function-wide unique Channel ID
  - An indication whether the audio channel is part of a ChannelGroup

Note: This specification also supports Ambisonic Clusters that have somewhat different Attributes. See Section 4.4, "Cluster Descriptor" for more details.

Each Cluster has an associated Cluster Descriptor that fully describes the Cluster.

A Cluster that is currently carrying valid audio information is called an Active Cluster. This specification also defines the Inactive Cluster, in which case the audio channels in the Cluster do not carry any audio and the audio content of the channels is effectively undefined.

Note: This is different from a Cluster that carries Silence data in its audio channels. Such a Cluster is still considered Active where audio is streaming, but the audio signals (samples or voltage levels) correspond to a zero audio level (muted audio).

The transition between Active and Inactive is always the result of a change within the Audio Function. The following situations may occur:

- The Output Cluster on an Entity's Output Pin may switch between Active and Inactive due to a change in Power State of the Power Domain to which the Entity belongs.
- The Output Cluster on a Selector Unit's Output Pin may switch between Active and Inactive depending on the position of the Unit's Selector Control.

- The Output Cluster on an Input Terminal's Output Pin may switch between Active and Inactive depending on whether audio data is present or not on the interface the Input Terminal represents. Likewise, a switch may occur depending on the presence of a valid clock on the Clock Input Pin of an Input Terminal.

Note that the Cluster Descriptor itself does not indicate whether the Cluster is Active or Inactive. The state of a Cluster (Active or Inactive) is always derived from another state in the Audio Function (see above). The Cluster's active state is exposed via a Read-Only Cluster Active Control, reflecting the current state of the Cluster (Active or Inactive). When a Cluster becomes Inactive, its Cluster Descriptor becomes undefined and shall not be used for any purpose. For example, an Entity that advertises a static output Cluster on its Output Pin may carry an Inactive Cluster at some point in time due to events that happen upstream in the audio path. In this case, although the Cluster Descriptor is still available and unchanged, the Cluster is Inactive, and the Cluster Descriptor shall not be used.

3.13.2.1 CHANNEL ID

Each channel in a Cluster shall be uniquely identifiable by its Channel ID. The Channel ID is a non-zero value in the **wChannelID** field in the channel's Information or Ambisonic Segment of the Cluster Descriptor. The Channel ID shall be Function-wide unique. The following rules apply:

- The Audio Function implementation shall assign a Channel ID to each channel entering the Audio Function via the Output Pin of an Input Terminal so that Host software has a means to trace individual channels as they traverse the Audio Function.
- Each Feature Unit, Effect Unit, or SRC Unit shall preserve the Channel IDs from its incoming Cluster to the outgoing Cluster.
- The Selector Unit shall preserve the Channel IDs from its currently selected incoming Cluster to the outgoing Cluster.
- A Mixer Unit, Processing Unit, or Extension Unit typically creates 'new' content in their outgoing Cluster, by applying some algorithm on any or all incoming channels. In this case, the Audio Function implementation should assign a new Channel ID to each channel in the outgoing Cluster. However, if there is a reason to indicate that the outgoing Cluster contains channel content predominantly originating from a specific incoming channel, then the Channel ID of that incoming channel may appear in the outgoing Cluster. It is left to the implementation to carefully consider how to populate the Channel IDs in the outgoing Cluster of these Unit types.

Subsequent sections provide more guidance on how to manage Channel IDs for all defined Terminal and Unit types.

3.13.3 INPUT TERMINAL

The Input Terminal (IT) is used to interface between the Audio Function's 'outside world' and other Units in the Audio Function. It serves as a receptacle for audio information flowing into the Audio Function. Its purpose is to represent a source of incoming audio data after this data has been properly extracted from the original audio stream into the separate logical channels that are embedded in this stream (the decoding process). The logical channels are grouped into a Cluster and leave the Input Terminal through its single Output Pin.

The Input Terminal that represents an audio stream entering the Audio Function by means of a USB OUT Endpoint, shall have a dedicated AudioStreaming OUT Interface that contains this Endpoint and there shall be a one-to-one relationship between that AudioStreaming Interface and its associated Input Terminal.

The Input Terminal may represent inputs to the Audio Function other than USB OUT Endpoints. A Line-In connector on an Audio Device is an example of such a non-USB input. A digital input connector, such as S/PDIF, is another example.

The Input Terminal Descriptor contains a field that either holds a direct reference to its associated AudioStreaming Interface or contains a list of Connector Entity IDs referencing its associated Connector Entities. The Host needs to use both the AudioStreaming Interface and Endpoint Descriptors or the Connector Entity Descriptors, in conjunction with the Input Terminal Descriptor to get a full understanding of the characteristics and capabilities of the Input Terminal. Stream-related parameters are stored in the AudioStreaming Interface or Connector Entity Descriptors. AudioControl-related parameters are stored in the Input Terminal Descriptor. Stream-related AudioControls are in the AudioStreaming Interface or the Connector Entity. Audio control-related AudioControls are in the Input Terminal.

The conversion process from incoming, possibly encoded, audio streams to logical audio channels always involves some form of decoding engine. The decoding types range from rather trivial decoding schemes like converting interleaved stereo 16-bit PCM data into Front Left and Front Right logical channels to very sophisticated schemes like converting an MPEG-2 7.1 encoded audio stream into Front Left, Front Left of Center, Front Center, Front Right of Center, Front Right, Back Left, Back Right and Low Frequency Effects logical channels. The decoding engine is considered part of the Entity that receives the encoded audio data streams (like a USB AudioStreaming Interface). The type of decoding is therefore implied by the value in the **wFormat** field, located in the class-specific AudioStreaming Self Descriptor. The associated Input Terminal deals with the logical channels after they have been decoded.

If there is an AudioStreaming Interface associated with the Input Terminal, then the Cluster Configuration on the Output Pin of the Input Terminal shall be determined through selection of the appropriate Alternate Setting of the associated AudioStreaming Interface. The Input Terminal shall have a Read-Only (r) Cluster Control, indicating which Cluster Configuration is currently in use. There shall be a one-to-one relationship between the Alternate Setting of the Interface and the corresponding Cluster Configuration as indicated by the CUR value of the Cluster Control. In other words, the reported CUR Cluster Control value shall be identical to the currently selected Alternate Setting in the associated AudioStreaming Interface. Also, the Cluster Configuration shall be compatible with the audio data format of the currently selected Alternate Setting of the AudioStreaming Interface. If Alternate Setting zero is currently selected, then the output Cluster shall be Inactive, and the reported CUR value of the Cluster Control shall be zero.

If there are one or more Connector Entities or internal transducers associated with the Input Terminal, then the Input Terminal shall have a Read-Write (rw) Cluster Control, used to select the Cluster Configuration on the Output Pin of the Input Terminal. It is the responsibility of the Host Software, potentially via user intervention, to choose a Cluster Configuration that is appropriate for the type of device that is currently plugged into the external Connector(s).

If the Input Terminal represents an embedded (set of) transducer(s), then the Cluster Control can be used to control the behavior of the transducer(s), such that they produce an audio stream that is compatible with the selected Cluster Configuration.

In both cases, the Input Terminal shall expose a Read-Only (r) Cluster Active Control that indicates whether the output Cluster is currently Active or Inactive.

The Audio Function shall assign unique Channel IDs to all channels in the output Cluster of the Input Terminal so that Host software has a means to trace individual channels as they traverse the Audio Function.

The Input Terminal has a single Clock Input Pin. The clock signal present at that Pin is used as the sampling clock for all underlying hardware that is represented by this Input Terminal. There is a field in the Input Terminal Descriptor that uniquely identifies the Clock Entity to which the Input Terminal is connected. If there is no need for the Audio Function to expose any clock information related to the Input Terminal, the Clock Input Pin of the Input Terminal may be left unconnected. In this case, it is assumed that the Input Terminal is internally connected to a clock that is always valid.

The Input Terminal optionally provides the Voltage Control. It is used to set the voltage level of the power supply that provides either phantom power or bias voltage to the Input Terminal's associated microphone.

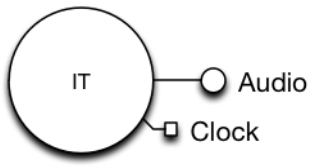
Depending on the state of the physical audio source the Input Terminal represents, the current Cluster on the Output Pin may be Inactive. For example, an AudioStreaming OUT Interface may currently be set to Alternate Setting 0 (non-streaming) and therefore, the Input Terminal that represents this Interface will have an Inactive output Cluster. Likewise, if the clock on the Input Terminal's Clock Input Pin is currently invalid, then the output Cluster shall be Inactive.

If the Input Terminal is an explicit member of a Power Domain, switching the Power Domain to any Power State other than PS0 or PS1 shall render the Input Terminal non-functional, and its output Cluster shall be Inactive.

In some cases, the Audio Function needs to indicate to Host software that the Input Terminal is functionally or physically related to one or more other Terminals or Entities. This can be expressed by using the Group construct. A typical example of such a relationship is one or more Input Terminals, representing a set of microphones, and an Output Terminal, representing the earpieces of a headset. They can be grouped together in a single Group that has these Terminals as its Members.

The symbol for the Input Terminal is depicted in the following figure:

Figure 3-6: Input Terminal Icon



3.13.4 OUTPUT TERMINAL

The Output Terminal (OT) is used to interface between Units inside the Audio Function and the 'outside world'. It serves as an outlet for audio information, flowing out of the Audio Function. Its purpose is to represent a sink of outgoing audio data before this data is properly packed from the original separate logical channels into the outgoing audio stream (the encoding process). The Cluster enters the Output Terminal through a single Input Pin. The Cluster Configuration on the Input Pin may change dynamically, depending on certain actions that occur on the upstream connection. This results in a change in the physical Cluster Configuration within the physical interface the Output Terminal represents.

An Output Terminal that represents an audio stream leaving the Audio Function by means of a USB IN Endpoint, shall have a dedicated AudioStreaming IN Interface that contains this Endpoint and there shall be a one-to-one relationship between that AudioStreaming Interface and its associated Output Terminal. In this case, when the Cluster on the Output Terminal's Input Pin changes, or if the clock to the Output Terminal becomes invalid, then the Audio Function shall switch the Active Alternate Setting of the AudioStreaming Interface to Alternate Setting 0

and update the Valid Alternate Settings Control to reflect which Alternate Settings are compatible with the new Cluster Configuration. The Host may then take appropriate action to start the stream.

If the Output Terminal is an explicit member of a Power Domain, switching the Power Domain to any Power State other than PS0 or PS1 shall render the Output Terminal non-functional, and the Audio Function shall switch the Active Alternate Setting of the AudioStreaming Interface to Alternate Setting 0 and update the Valid Alternate Settings Control to reflect that none of the Alternate Settings are currently valid.

The Output Terminal may represent outputs from the Audio Function other than USB IN Endpoints. A speaker built into an Audio Device, or a Line-Out connector is an example of such a non-USB output. For an Output Terminal that does not represent an AudioStreaming Interface, handling of Cluster Configuration changes at the Output Terminal's Input Pin is left to the Audio Function implementation.

The Output Terminal Descriptor contains a field that either holds a direct reference to its associated AudioStreaming Interface or contains a list of Connector Entity IDs referencing its associated Connector Entities or internal transducers. The Host needs to use both the AudioStreaming Interface and Endpoint Descriptors or the Connector Entity Descriptors, in conjunction with the Output Terminal Descriptor to get a full understanding of the characteristics and capabilities of the Output Terminal. Stream-related parameters are stored in the AudioStreaming Interface or Connector Entity Descriptors. AudioControl-related parameters are stored in the Output Terminal Descriptor. Stream-related AudioControls are in the AudioStreaming Interface or the Connector Entity. Audio control-related AudioControls are in the Output Terminal.

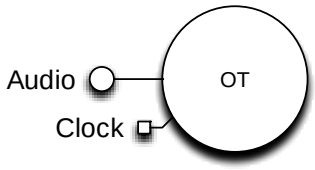
The conversion process from incoming logical audio channels to possibly encoded audio streams always involves some form of encoding engine. The encoding engine is considered part of the Entity that transmits the encoded audio data streams (like the AudioStreaming Interface). The type of encoding is therefore implied by the value in the **wFormat** field, located in the class-specific AudioStreaming Self Descriptor. The associated Output Terminal deals with the logical channels before encoding.

The Output Terminal has a single Clock Input Pin. The clock signal present at that Pin is used as the sampling clock for all underlying hardware that is represented by this Output Terminal. There is a field in the Output Terminal Descriptor that uniquely identifies the Clock Entity to which the Output Terminal is connected. If there is no need for the Audio Function to expose any clock information related to the Output Terminal, the Clock Input Pin of the Output Terminal may be left unconnected. In this case, it is assumed that the Output Terminal is internally connected to a clock that is always valid.

In some cases, the Audio Function needs to indicate to Host software that the Output Terminal is functionally or physically related to one or more other Terminals or Entities. This can be expressed by using the Group construct. A typical example of such a relationship is one or more Input Terminals, representing a set of microphones, and an Output Terminal, representing the earpieces of a headset. They can be grouped together in a single Group that has these Terminals as its Members.

The symbol for the Output Terminal is depicted in the following figure:

Figure 3-7: Output Terminal Icon



3.13.4.1 EN 50332 SUPPORT

This specification supports both the EN 50332-2:2013 and the EN 50332-3:2017 standards. Both are aimed to limit the amount of exposure to high SPL levels, typically when using headphones or headsets for extended periods of time. The EN 50332-2 standard describes a method of passively reporting acoustic or electrical output levels whereas the EN 50332:3 standard provides a more dynamic dosage reporting mechanism where the Device actively calculates actual dosage levels on an ongoing basis. For details, refer to the appropriate specifications [EN 50332-2:2013] and [EN 50332-2:2017]. A Device is allowed to support either EN 50332-2 or EN 50332-3 specifications. However, if the Device supports EN 50332-3, it is recommended the Device also supports EN 50332-2.

3.13.4.1.1 EN 50332-2

When the Device supports the EN 50332-2:2013 specification, it exposes either an EN 50332-2 Acoustic Level Segment or an EN 50332-2 Voltage Level Segment as part of its Terminal Companion Descriptor for each Output Terminal that supports the EN 50332-2 specification. (For details, see Section 4.5.3.4.3.2.1, “EN 50332-2 Acoustic level Segment” and Section 4.5.3.4.3.2.2, “EN 50332-2 Voltage level Segment.”) Host software can then use this information to gauge and potentially limit dosage and exposure levels to the user.

3.13.4.1.2 EN 50332-3

When the Device supports the EN 50332-3:2017 specification, it exposes a Momentary Exposure Level (MEL) Control in the Output Terminal that supports the specification. The MEL Control periodically reports back to the Host the current Exposure Level, according to the following rules:

Every second, the USB Audio Device shall compute the Momentary Exposure Level based on all of the audio samples feeding the digital-to-analog converters within that second. The MEL Control shall generate an interrupt message within 2 interrupt endpoint Service Intervals of the new MEL value becoming available. The intent is for the Host to read the MEL Control in response to each interrupt. The Host is then responsible for utilizing the MEL values to perform additional calculations and threshold comparisons, culminating in the display of appropriate warnings and/or the reduction of the output level in accordance with EN 50332-3:2017 and related standards.

To compute Momentary Exposure Level (MEL) for headphones or a headphone jack, a Device shall perform the following computations every second:

1. Apply an A-weighting filter to the stereo digital audio signal that is sent to the digital-to-analog converter.
2. To the result of step 1, apply a transfer function representing the digital-to-analog converter and amplifier’s frequency response and output gain, resulting in a stereo audio signal expressed in volts (V).
    - If the output is a headphone speaker, apply a transfer function to the result representing the headphone speaker’s frequency response and sensitivity, assuming a Head And Torso Simulator (HATS) diffuse field measurement.

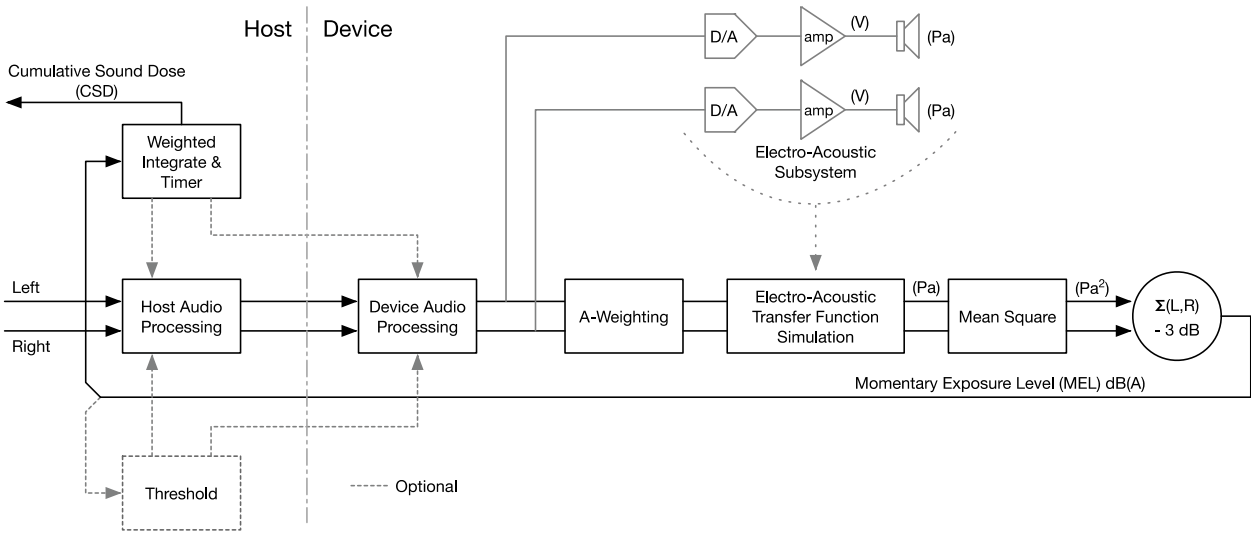
- If the output is a headphone jack, assume a default sensitivity (as derived from the relevant tables in the EN 50332-3 specification) for an unknown set of attached headphones by multiplying by 40/3 Pa/V.

This will result in a stereo audio signal expressed in pascals (Pa).

3. To each audio channel of the result of step 2, square all the samples within this 1-second interval, sum them, and divide by the number of samples per audio channel within the 1-second interval, producing two "mean square" numbers expressed in pascals squared (Pa^2).
4. Using the results of step 3, add the two "mean square" numbers, divide by the square of 20 µPa, take the base-10 logarithm, multiply by 10 to yield dB, and finally subtract 3 dB (as per the EN 50332-3 specification). This produces a single MEL value expressed in dB(A).

The following figure further illustrates the procedure outlined above:

Figure 3-8: MEL Procedure



3.13.5 MIXER UNIT

The Mixer Unit (MU) transforms two or more logical input channels into one or more logical output channels. The input channels are grouped into one or more Clusters. Each Cluster enters the Mixer Unit through one of the p Input Pins. The logical output channels are grouped into one Cluster and leave the Mixer Unit through a single Output Pin.

The Pin channels on each Input Pin i of the Mixer Unit are numbered from one to N_i. The PCC value N_i for each of the Input Pins is inherited from the first upstream Entity that defines a PCC value on its Output Pin.

Every input channel can be mixed into all the output channels. If N is the total number of Input Pin channels ($N = \sum_{i=1}^{p} N_i$) and M is the number of Output Pin channels the Mixer supports, then there is a two-dimensional array ($N \times M$) of Mixer Controls in the Mixer Unit. Some Mixer Controls may be Read-Only and have fixed values. For example, a 'missing' connection may be advertised as a Read-Only Mixer Control with fixed value -∞ dB.

On each Input Pin i, the incoming Cluster logical channels, numbered from one to n_i, are mapped one-to-one onto the Pin channels on that Input Pin, i.e. for each Input Pin i, Cluster channel 1 is mapped onto Pin channel 1, Cluster channel 2 is mapped onto Pin channel 2, and so on, up to Cluster channel n_i which is mapped onto Pin channel n_i. Note that the PCC values on each of the Mixer Unit's Input Pins (N_i) may be different from the actual number of channels in the incoming Clusters (n_i). However, for each Input Pin i, $N_i \geq n_i$ shall always be true.

Mixer Controls residing on currently unused Pin channels shall always remain accessible and retain their last setting for as long as the Mixer Unit remains in a Power State that requires this.

The Cluster Control is typically implemented as Read (r) and exposes a single Cluster Configuration. However, the Mixer Unit can expose multiple Cluster Configurations on its Output Pin and implement the Cluster Control as Read-Write (rw). A Cluster Control value of zero indicates that the output Cluster Configuration is inherited from the current Cluster on Input Pin one (the dominant Input Pin). The Read-Only Cluster Active Control shall always be implemented and indicates whether the output Cluster is currently Active or not.

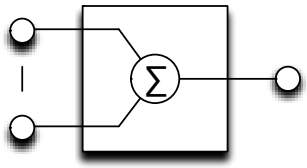
Since the Mixer Unit likely redefines the channels in its output Cluster as a processed result of any of the incoming channels, it would be appropriate to treat these channels as independent of all other channels in the Audio Function and therefore assign them unique IDs in their respective **wChannelID** fields in the output Cluster. However, this is not a requirement, and it is left to the implementation to assign Channel ID values as accurately as possible. For example, a Mixer Unit that has two inputs, each carrying a stereo Cluster, where the Cluster on Input Pin one is the main audio stream and the Cluster on Input Pin two is mixed into that main audio stream, it may be appropriate to propagate the Channel IDs from the Cluster on Input Pin one to the output Cluster.

The Mixer Unit can also be used to repurpose or redefine the channel relationships in a Cluster. By creating a Mixer Unit with a single Input Pin, and setting up the proper Mixer Controls, it can freely define its Output Cluster and its Descriptor to serve the intended purpose. For example, an incoming stereo Cluster that is defined as Front Left, Front Right, can be redefined to become Headphone Left, Headphone Right, by setting the Mixer Controls such that the Front Left and Front Right channels get fully mixed into output channels one and two respectively and by setting the **wRelationship** fields of the output Cluster Descriptor to Headphone Left and Headphone Right, respectively. As another example, the **wPurpose** fields of certain channels could be modified to reflect the actual purpose of the output Cluster of the Mixer Unit. This could be used to create a Cluster specifically designated for ultrasonic purposes, starting from an incoming Cluster that contains full bandwidth information, including ultrasonic information.

If the Mixer Unit is an explicit member of a Power Domain, switching the Power Domain to any Power State other than PS0 or PS1 shall render the Unit non-functional, and its output Cluster shall be Inactive.

The symbol for the Mixer Unit can be found in the following figure:

Figure 3-9: Mixer Unit Icon



3.13.6 SELECTOR UNIT

The Selector Unit (SU) has p Input Pins and a single Output Pin. It incorporates a Selector Control that allows to select one Cluster from the p Clusters on the Input Pins, and routes that Cluster unaltered to the single Output Pin. It represents a multi-channel source selector, capable of selecting between p sources where p shall be larger than one. Note that each incoming Cluster may have different characteristics in terms of the number of channels, their Purpose, Relationship, Grouping, or Channel IDs. To determine the current Cluster on the Output Pin, Host software needs to use the currently selected Input Pin and track the audio path upstream on that Input Pin until it

finds an Entity's Output Pin that advertises an Output Cluster definition and its Active State (the Selector Unit itself does not contain a Cluster Active Control).

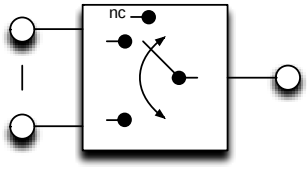
Each of the Input Pins inherits the PCC value N_p from the first upstream Entity that defines a PCC value on its Output Pin. The Selector Unit does not advertise a PCC value on its output. It is the maximum of all PCC values on its Input Pins.

The Selector Unit also has the optional capability to disconnect the output from all its inputs. In this case the Cluster on the Output Pin does not contain audio and is therefore an Inactive Cluster. If the disconnect option is supported, the range of the Selector Control CUR Attribute shall include zero.

If the Selector Unit is an explicit member of a Power Domain, switching the Power Domain to any Power State other than PS0 or PS1 shall render the Unit non-functional, and its output Cluster shall be Inactive.

The symbol for the Selector Unit can be found in the following figure:

Figure 3-10: Selector Unit Icon



3.13.7 FEATURE UNIT

The Feature Unit (FU) is essentially an N-channel Unit that provides basic manipulation of a (cascaded) set of single-parameter audio processes on the incoming logical channels. For each Pin channel, the Feature Unit optionally provides AudioControls for the following features:

- Bypass
- Mute
- Gain
- Tone Control (Bass, Mid, Treble)
- Graphic Equalizer
- Automatic Gain Control
- Delay
- Bass Boost
- Loudness
- Input Gain Pad
- Phase Inverter

In addition, the Feature Unit optionally provides the above AudioControls but now influencing all channels at once. In this way, Cluster-wide AudioControls can be implemented. The Cluster-wide AudioControls are cascaded after the individual channel AudioControls. This setup is especially useful in multi-channel systems where the individual channel AudioControls may be used for channel balancing and the Cluster-wide AudioControls may be used for overall settings.

A Feature Unit shall implement at least one AudioControl.

The Pin channels in the Feature Unit are numbered from one to N. The PCC value N is inherited from the first upstream Entity that defines a PCC value on its Output Pin. The Primary channel has channel number zero and is always virtually present.

The Feature Unit Descriptor reports which AudioControls are present for every Pin channel in the Feature Unit, including the Primary channel. All Pin channels in the Feature Unit are fully independent. There exists no cross coupling among channels within the Feature Unit.

The Feature Unit never alters the incoming Cluster in terms of the number of logical channels in the Cluster, n, nor their Purpose, Relationship, Grouping, or Channel IDs. In other words, the Feature Unit does not in any way redefine the incoming Cluster description and there are always as many logical output Cluster channels as there are input Cluster channels. The Cluster enters the Feature Unit through a single Input Pin and leaves the Unit through a single Output Pin.

The incoming logical Cluster channels are mapped one-to-one onto the Pin channels of the Feature Unit, i.e., Cluster channel 1 is mapped onto Pin channel 1, Cluster channel 2 is mapped onto Pin channel 2, and so on, up to Cluster channel n which is mapped onto Pin channel n. Note that the number of Pin channels (N) in the Feature Unit may be different from the actual number of logical channels in the incoming Cluster (n). However, $N \geq n$ shall always be true.

AudioControls residing on currently unused Pin channels shall always remain accessible and retain their last setting for as long as the Feature Unit remains in a Power State that requires this.

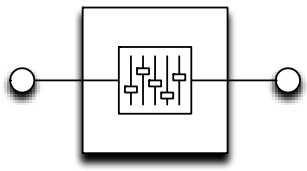
If the optional Bypass Control is present on a Pin channel, then engaging the bypass function shall result in passing the corresponding logical channel of the incoming Cluster unaltered to that same logical channel of the outgoing Cluster.

If the optional Bypass Control is present on the Primary channel, then engaging the bypass function shall result in passing the incoming Cluster unaltered to the output of the Unit.

If the Feature Unit is an explicit member of a Power Domain, switching the Power Domain to any Power State other than PS0 or PS1 shall render the Unit non-functional, and its output Cluster shall be Inactive.

The symbol for the Feature Unit is depicted in the following figure:

Figure 3-11: Feature Unit Icon



3.13.8 SAMPLING RATE CONVERTER UNIT

The Sampling Rate Converter (SRC) Unit (RU) is included here as an optional way to indicate where exactly within the Audio Function sampling rate conversion takes place. In many cases, it is unnecessary to indicate this point and any SRC Unit may be omitted from the topology without materially affecting the information presented to the Host. The primary reason to include the SRC Unit is to accurately report any latencies incurred by the Sampling Rate Conversion process.

The SRC Unit provides a bridge function between different Clock Domains within the Audio Function. The SRC Unit does not provide AudioControls that impact the SRC functionality of the Unit. It takes the audio on all the logical channels in the input Cluster belonging to a certain Clock Domain and converts them into the same logical channels in the output Cluster but now belonging to another Clock Domain.

The SRC Unit never alters the incoming Cluster in terms of the number of logical channels in the Cluster, n, nor their Purpose, Relationship, Grouping, or Channel IDs. In other words, the SRC Unit does not in any way redefine the incoming Cluster description and there are always as many logical output Cluster channels as there are input Cluster channels. The Cluster enters the SRC Unit through a single Input Pin and leaves the Unit through a single Output Pin.

The Pin channels in the SRC Unit are numbered from one to N. The PCC value N is inherited from the first upstream Entity that defines a PCC value on its Output Pin. There is no Primary channel.

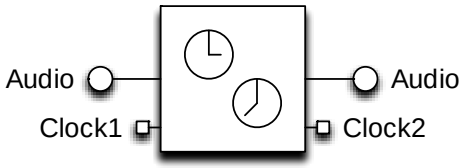
The incoming logical Cluster channels are mapped one-to-one onto the Pin channels of the SRC Unit, i.e., Cluster channel 1 is mapped onto Pin channel 1, Cluster channel 2 is mapped onto Pin channel 2, and so on, up to Cluster channel n which is mapped onto Pin channel n. Note that the number of Pin channels (N) in the SRC Unit may be different from the actual number of logical channels in the incoming Cluster (n). However, $N \geq n$ shall always be true.

The SRC Unit has two Clock Input Pins. One Clock Input Pin is associated with the single Input Pin of the SRC Unit. The other Clock Input Pin is associated with the single Output Pin of the SRC Unit. The clock signals present at those two Clock Input Pins identify the two Clock Domains between which the SRC Unit is converting. Note that it is allowed to have both Clock Input Pins connected to clock signals belonging to the same Clock Domain. It is also allowed to leave one or both Clock Input Pins unconnected if there is no need for the Audio Function to expose clock information related to the unconnected side of the SRC Unit.

If the SRC Unit is an explicit member of a Power Domain, switching the Power Domain to any Power State other than PS0 or PS1 shall render the Unit non-functional, and its output Cluster shall be Inactive.

The symbol for the SRC Unit is depicted in the following figure:

Figure 3-12: Sampling Rate Converter Unit Icon



3.13.9 EFFECT UNIT

The Effect Unit (EU) is a multi-channel processing unit that provides advanced manipulation of a multi-parameter AudioControl on the incoming logical channels on a per-channel basis. For each logical channel, the Effect Unit shall provide one of the following AudioControls:

- Parametric Equalizer Section
- Reverberation
- Modulation Delay
- Dynamic Range Compressor

In addition, the Effect Unit optionally provides one of the above AudioControls but now influencing all channels of the Cluster at once. In this way, a Cluster-wide AudioControl may be implemented. The Cluster-wide AudioControl is cascaded after the individual channel AudioControls. This setup is especially useful in multi-channel systems where the individual channel AudioControls may be used for channel balancing and the Cluster-wide AudioControl may be used for overall settings.

The Pin channels in the Effect Unit are numbered from one to N. The PCC value N is inherited from the first upstream Entity that defines a PCC value on its Output Pin. The Primary channel has channel number zero and is always virtually present.

The Effect Unit Descriptor reports which AudioControls are present for every Pin channel in the Effect Unit, including the Primary channel. All Pin channels in the Effect Unit are fully independent. There exists no cross coupling among channels within the Effect Unit.

The Effect Unit never alters the incoming Cluster in terms of the number of logical channels in the Cluster, n, nor their Purpose, Relationship, Grouping, or Channel IDs. In other words, the Effect Unit does not in any way redefine the incoming Cluster description and there are always as many logical output Cluster channels as there are input Cluster channels. The Cluster enters the Effect Unit through a single Input Pin and leaves the Unit through a single Output Pin.

The incoming logical Cluster channels are mapped one-to-one onto the Pin channels of the Effect Unit, i.e., Cluster channel 1 is mapped onto Pin channel 1, Cluster channel 2 is mapped onto Pin channel 2, and so on, up to Cluster channel n which is mapped onto Pin channel n. Note that the number of Pin channels (N) in the Effect Unit may be different from the actual number of logical channels in the incoming Cluster (n). However, $N \geq n$ shall always be true.

AudioControls residing on currently unused Pin channels shall always remain accessible and retain their last setting for as long as the Effect Unit remains in a Power State that requires this.

If the optional Bypass Control is present on a Pin channel, then engaging the bypass function shall result in passing the corresponding logical channel of the incoming Cluster unaltered to that same logical channel of the outgoing Cluster.

If the optional Bypass Control is present on the Primary channel, then engaging the bypass function shall result in passing the incoming Cluster unaltered to the output of the Unit.

If the Effect Unit is an explicit member of a Power Domain, switching the Power Domain to any Power State other than PS0 or PS1 shall render the Unit non-functional, and its output Cluster shall be Inactive.

3.13.9.1 PARAMETRIC EQUALIZER SECTION EFFECT UNIT

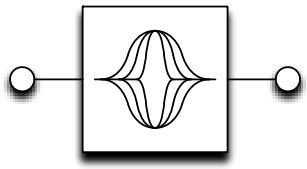
The Parametric Equalizer Section (PE) Effect Unit is used to manipulate and equalize the frequency characteristics of the original audio information around a certain center frequency. To build a parametric equalizer, a number of these PEQS Effect Units may need to be cascaded to obtain the desired functionality. The parameters that can be manipulated to obtain the desired equalizing effect are:

- Center Frequency: the frequency around which the audio spectrum is manipulated. Expressed in Hz.
- Q Factor: a measure for the range of frequencies around the center frequency that are influenced. Expressed as a ratio.
- Gain: the amount of gain or attenuation at the center frequency. Expressed in dB.

The algorithm to produce the desired equalization effect can be manipulated on a per-channel basis. The Primary channel concept allows equalization for all channels simultaneously.

The symbol for the PE Processing Unit can be found in the following figure:

Figure 3-13: PEQS Effect Unit Icon



3.13.9.2 REVERBERATION EFFECT UNIT

The Reverberation (RV) Effect Unit is used to add room acoustics effects to the original audio information. These effects may range from small room reverberation effects to simulation of a large concert hall reverberation. A set of AudioControls can be manipulated to obtain the desired reverberation effects.

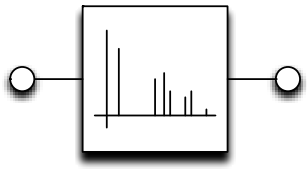
- Reverb Type: Room1 (small room), Room2 (medium room), Room3 (large room), Hall1 (medium concert hall), Hall2 (large concert hall), Plate, Delay, and Panning Delay. This is a macro Control that chooses among potentially different algorithms to obtain the desired reverberation effect. Changing this Control has no impact on the value of the other AudioControls below. However, Host software may choose to reprogram the Reverb Time Control (as an example) to a default value for the selected Reverb Type when the Reverb Type Control is set to a different Reverb Type.
- Reverb Level: sets the amount of reverberant sound versus the original sound. Expressed as a ratio.
- Reverb Time: sets the time over which the reverberation will continue. Expressed in s.
- Reverb Delay Feedback: used with Reverb Types Delay and Delay Panning. Sets the way in which delay repeats. Expressed as a ratio.
- Reverb Pre-Delay: sets the delay time between original sound and initial reverb reflection. Expressed in ms.
- Reverb Density: sets the density of the reverb reflections.
- Reverb Hi-Freq Roll-Off: sets the cut-off frequency of a low pass filter on the reflections. Expressed in Hz.

It is entirely left to the designer how a certain reverberation effect is obtained. It is not the intention of this specification to precisely define all the parameters that influence the reverberation experience (for instance in a multi-channel system, it is possible to create very similar reverberation impressions, using different algorithms and parameter settings on all channels).

The algorithm to produce the desired equalization effect can be manipulated on a per-channel basis. The Primary channel concept allows equalization for all channels simultaneously.

The symbol for the Reverberation Effect Unit can be found in the following figure:

Figure 3-14: Reverberation Effect Unit Icon



3.13.9.3 MODULATION DELAY EFFECT UNIT

The Modulation Delay (MD) Effect Unit is used to add modulation (like chorus) effects to the original audio information. A number of parameters can be manipulated to obtain the desired modulation effects.

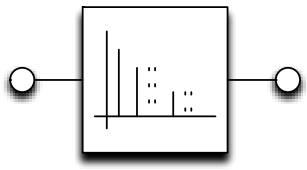
- Modulation Delay Balance: controls the ratio of the original sound to that of the effected sound. Expressed as a ratio.
- Modulation Delay Rate: sets the speed (frequency) of the modulator. Expressed in Hz.
- Modulation Delay Depth: sets the depth at which the sound is modulated. Expressed in ms.
- Modulation Delay Time: sets the delay that is added to the modulated sound before adding it to the original sound. Expressed in ms.
- Modulation Delay Feedback Level: controls the amount of the modulated sound that is routed back to the input of the modulator unit. Expressed as a ratio.

It is entirely left to the designer how a certain modulation effect is obtained.

The algorithm to produce the desired equalization effect can be manipulated on a per-channel basis. The Primary channel concept allows equalization for all channels simultaneously.

The symbol for the Modulation Delay Effect Unit can be found in the following figure:

Figure 3-15: Modulation Delay Effect Unit Icon

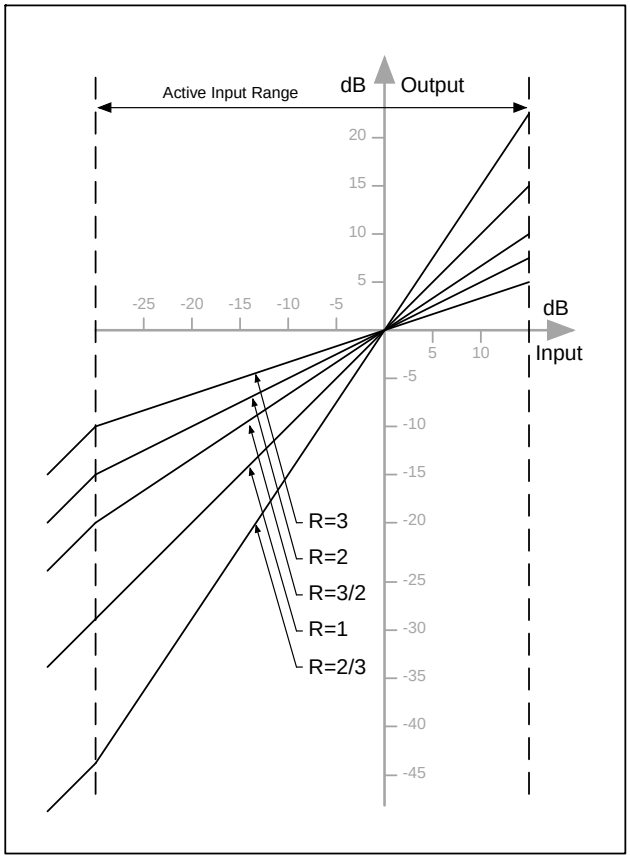


3.13.9.4 DYNAMIC RANGE COMPRESSOR/EXPANDER EFFECT UNIT

The Dynamic Range Compressor/Expander (DR) Effect Unit is used to intelligently compress, expand, or limit the dynamic range of the original audio information. Several parameters can be manipulated to influence the desired effect.

Note: Two Dynamic Range Compressor/Expander Effect Units may be used together for companding.

Figure 3-16: Dynamic Range Compressor/Expander Transfer Characteristic



- Ratio R: determines the slope of the static input-to-output transfer characteristic in the effect's active input range. The effect is defined in terms of the ratio R, which is the inverse of the derivative of the output power P_O as a function of the input power P_I when P_O and P_I are expressed in dB.

$$R^{-1} = \frac{\partial Log(\frac{P_O}{P_R})}{\partial Log(P_I/P_R)}$$

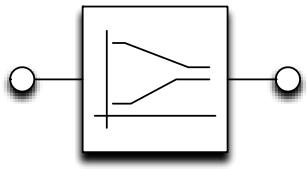
  P_R is the reference level, and it is made equal to the so-called line level. All levels are expressed relative to the line level (0 dB), which is usually 15-20 dB below the maximum level. Compression is obtained when R > 1, R = 1 does not affect the signal and R < 1 gives rise to expansion.
- Maximum Amplitude: the upper boundary of the active input range, relative to the line level (0 dB). Expressed in dB.
- Threshold level: the lower boundary of the active input level, relative to the line level (0 dB).
- Attack Time: determines the response of the effect as a function of time to a step in the input level. Expressed in ms.
- Release Time: relates to the recovery time of the gain of the compressor after audio is no longer within the boundaries between Threshold and Maximum Amplitude. Expressed in ms.
- Make-up Gain: set to compensate for the gain loss in the effect. Expressed in dB.

It is entirely left to the designer how a certain dynamic range effect is obtained.

The algorithm to produce the desired equalization effect can be manipulated on a per-channel basis. The Primary channel concept allows equalization for all channels simultaneously.

The symbol for the Dynamic Range Compressor/Expander Effect Unit can be found in the following figure:

Figure 3-17: Dynamic Range Compressor/Expander Effect Unit Icon



3.13.10 PROCESSING UNIT

The Processing Unit (PU) represents a functional block inside the Audio Function that transforms one or more logical input channels into one or more logical output channels. The input channels are grouped into one or more Clusters. Each Cluster enters the Processing Unit through one of the p Input Pins. The logical output channels are grouped into one Cluster and leave the Processing Unit through a single Output Pin.

The Pin channels on each Input Pin i of the Processing Unit are numbered from one to N_i. The PCC value N_i for each of the Input Pins is inherited from the first upstream Entity that defines a PCC value on its Output Pin.

On each Input Pin i, the incoming Cluster logical channels, numbered from one to n_i, are mapped one-to-one onto the Pin channels on that Input Pin, i.e. for each Input Pin i, Cluster channel 1 is mapped onto Pin channel 1, Cluster channel 2 is mapped onto Pin channel 2, and so on, up to Cluster channel n_i which is mapped onto Pin channel n_i. Note that the number of Pin channels on each of the Processing Unit's Input Pins (N_i) may be different from the actual number of channels in the incoming Clusters (n_i). However, for each Input Pin i, $N_i \geq n_i$ shall always be true.

AudioControls residing on currently unused channels shall always remain accessible and retain their last setting for as long as the Processing Unit remains in a Power State that requires this.

This specification defines several standard transforms (algorithms) that are considered necessary to support additional Audio functionality; these transforms are not covered by the other Unit types but are commonplace enough to be included in this specification so that a generic driver can provide control for it.

A Processing Unit shall implement at least one of the defined algorithms.

If the optional Bypass Control is present, then engaging the bypass function shall result in passing the incoming Cluster on Input Pin one (the dominant Input Pin) unaltered to the output of the Unit. If it is necessary to be able to bypass the Processing Unit's functionality but providing an output Cluster different from the input Cluster on Input Pin one, then an explicit bypass topology using a Selector Unit should be implemented.

The Processing Unit can expose multiple Cluster Configurations on its Output Pin. The Cluster Control is typically implemented as Read-Write (rw) and is used by Host software to select the desired Cluster Configuration on the Output Pin. In other words, the selection of the Cluster Configuration controls the operational mode of the Processing Unit. A Cluster Control value of zero indicates that the output Cluster Configuration is inherited from the current Cluster on Input Pin one (the dominant Input Pin). The Read-Only Cluster Active Control Shall always be implemented and indicates whether the output Cluster is currently Active or not.

Generally, the Processing Unit redefines the channels in its output Cluster as a processed result of any or all of the incoming channels. It is therefore appropriate to treat these channels as independent of all other channels in the Audio Function and assign them unique IDs in their respective **wChannelID** fields in the output Cluster. However, in subsequent sections, some guidelines are provided on how to manage Channel IDs for the different types of Processing Units, defined by this specification.

If the Processing Unit is an explicit member of a Power Domain, then switching the Power Domain to any Power State other than PS0 or PS1 shall render the Unit non-functional, and its output Cluster shall be Inactive.

3.13.10.1 UP/DOWN-MIX PROCESSING UNIT

The Up/Down-mix (UD) Processing Unit provides facilities to derive *m* output audio channels from *n* input audio channels. The algorithms and transforms applied to accomplish this are not defined by this specification and may be proprietary. The input channels are grouped into one or more Clusters. Each Cluster enters the Processing Unit through one of the p Input Pins. The logical output channels are grouped into one Cluster and leave the Up/Down-mix Processing Unit through a single Output Pin.

The Up/Down-mix Processing Unit may support multiple modes of operation. The logical input channels in the incoming Clusters are defined by Entities in the upstream audio path to which the Input Pins of the Up/Down-mix Processing Unit are connected. The Up/Down-mix Processing Unit Descriptor reports which up/down-mixing modes the Unit supports through its **waClusterDescrID()** array. Each element of the **waClusterDescrID ()** array indicates which output channels in the output Cluster are effectively present in a particular mode. Mode selection is accomplished by selecting an output Cluster through the Cluster Control.

As an example, consider the case where an Up/Down-mix Processing Unit is connected to the Input Terminal, producing Dolby™ AC-3 5.1 decoded audio. The input Cluster to the Up/Down-mix Processing Unit therefore contains Front Left, Front Right, Front Center, Surround Array Left, Surround Array Right (Left Surround and Right Surround in CEA-861.2 parlance), and LFE logical channels.

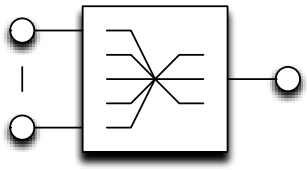
Suppose the Audio Function's hardware is limited to reproducing only dual channel audio. Then the Up/Down-mix Processing Unit could use some (sophisticated) algorithms to down-mix the available spatial audio information into two ('enriched') channels so that the maximum spatial effects can be experienced, using only two channels. It is left to the implementation to use the appropriate down-mix algorithm depending on the physical nature of the Output Terminal to which the Up/Down-mix Processing Unit is eventually routed. For instance, a different down-mix algorithm may be needed whether the 'enriched' stereo stream is sent to a pair of speakers or to a headset. However, this knowledge already resides within the Audio Function and deciding which down-mix algorithm to use does not need Host intervention.

As a second interesting example, suppose the hardware is capable of servicing eight discrete audio channels (for example, a full-fledged MPEG-2 7.1 system). Now the Up/Down-mix Processing Unit could use certain techniques to derive meaningful content for the extra audio channels (Front Left of Center, Front Right of Center) that are present in the output Cluster and are missing in the input channel Cluster (AC-3 5.1). This is a typical example of an up-mix situation.

Since the Up/Down-mix Processing Unit redefines the channels in its output Cluster as a processed result of any or all of the incoming channels, it is appropriate to treat these channels as independent of all other channels in the Audio Function and therefore assign them unique IDs in their respective **wChannelID** fields in the output Cluster.

The symbol for the Up/Down-mix Processing Unit is depicted in the following figure:

Figure 3-18: Up/Down-mix Processing Unit Icon



3.13.10.2 CHANNEL REMAP PROCESSING UNIT

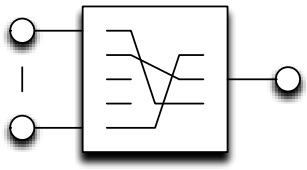
The Channel Remap (CR) Processing Unit provides facilities to derive *m* output audio channels from *n* input audio channels. It creates an output Cluster that contains a subset of the input channels, effectively acting as a simple channel selection mechanism with no audio processing involved. The input channels are grouped into one or more Clusters. Each Cluster enters the Processing Unit through one of the p Input Pins. The logical output channels are grouped into one Cluster and leave the Channel Remap Processing Unit through a single Output Pin.

The logical input channels in the incoming Clusters are defined by Entities in the upstream audio path to which the Input Pins of the Channel Remap Processing Unit are connected. The Channel Remap Processing Unit may support multiple channel remapping modes. The Channel Remap Processing Unit Descriptor reports which channel remapping modes the Unit supports through its **waClusterDescrID()** array. Each element of the **waClusterDescrID()** array indicates which output channels in the output Cluster are effectively present in a particular remapping mode. Mode selection is accomplished by selecting an output Cluster through the Cluster Control.

Since the Channel Remap Processing Unit does not redefine the channels in its output Cluster but merely selects channels from the incoming channels, it shall preserve the Channel IDs associated with those incoming channels as they are bundled to create the outgoing Cluster.

The symbol for the Channel Remap Processing Unit is depicted in the following figure:

Figure 3-19: Channel Remap Processing Unit Icon



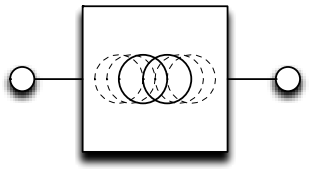
3.13.10.3 STEREO EXTENDER PROCESSING UNIT

The Stereo Extender (ST) Processing Unit operates on Front Left and Front Right channels only. It processes the Front Left and Right from the incoming Cluster to expand the sound field and to make it appear to originate from outside the Front Left/Right speaker locations. Extended stereo effects may be achieved via various methods. All other channels in the Cluster are passed through without modification. The algorithms and transforms applied to accomplish the stereo widening are not defined by this specification and may be proprietary. The perceived width of the sound field can be controlled via the Width Control.

Since the Stereo Extender Processing Unit does not redefine any of the channels in its output Cluster, it shall preserve the Channel IDs associated with those incoming channels.

The symbol for the Stereo Extender Unit is depicted in the following figure:

Figure 3-20: Stereo Extender Processing Unit Icon



3.13.10.4 MULTI-FUNCTION PROCESSING UNIT

The Multi-Function (MF) Processing Unit groups different but related algorithmic blocks that together provide a certain functionality. This specification defines several algorithms that are considered useful and commonplace enough to be included in this specification.

The mandatory Read-Only (r) AlgoPresent Control returns a bitmap indicating what types of algorithms are performed inside the Multi-Function Processing Unit. The following algorithms are currently supported:

- Algorithm Undefined
- Beam Forming Algorithm
- Acoustic Echo Cancellation Algorithm
- Active Noise Cancellation Algorithm
- Blind Source Separation Algorithm
- Noise Suppression/Reduction

The exact implementation of these algorithms and how they may interact is implementation dependent.

Note 1: If there is a need to expose to the Host how the algorithms are interconnected, a designer may choose to model the assembly of algorithms using multiple Multi-Function Processing Units, each containing just one or a subset of algorithms and explicitly connecting them together.

Note 2: Most of the algorithms mentioned above involve some form of signal processing that cannot be assumed to be linear and time invariant.

Note that support for an algorithm may change dynamically (due to Audio Function resource reallocation, for example). In this case, the AlgoPresent Control shall reflect the new situation and generate an interrupt to inform the Host of the change.

The optional AlgoEnable Control allows to selectively enable or disable the implemented algorithms.

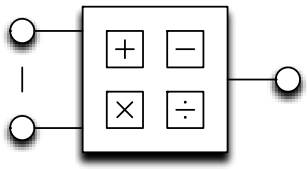
During normal operation, the Multi-Function Processing Unit transforms one or more logical input channels into one or more logical output channels. The input channels are grouped into one or more Clusters. Each Cluster enters the Processing Unit through one of the p Input Pins. The logical output channels are grouped into one Cluster and leave the Processing Unit through a single Output Pin.

Since the Multi-Function Processing Unit redefines the channels in its output Cluster as a processed result of any or all the incoming channels, it is appropriate to treat these channels as independent of all other channels in the Audio Function and therefore assign them unique IDs in their respective **wChannelID** fields in the output Cluster.

A Multi-Function Processing Unit shall implement the Bypass Control so that a generic audio driver that does not understand what functionality is implemented in the Multi-Function Processing Unit will be capable of removing it from the signal path.

The symbol for the Multi-Function Processing Unit is depicted in the following figure:

Figure 3-21 Multi-Function Processing Unit Icon



3.13.11 EXTENSION UNIT

The Extension Unit (XU) is the method provided by this specification to easily add vendor-specific building blocks to the specification. The functionality and behavior of the Extension Unit is identified by a Globally Unique Identifier (GUID) that is included in the Extension Unit Descriptor.

Note: This GUID is not used to identify *instances* of a Device. Rather, the same GUID is used in all implementations that incorporate this Extension Unit with this functionality and behavior, requiring the same vendor-defined software to operate.

The Extension Unit provides vendor-defined functionality inside the Audio Function that transforms one or more logical input channels into one or more logical output channels. The input channels are grouped into one or more Clusters. Each Cluster enters the Extension Unit through one of the p Input Pins. The logical output channels are grouped into one Cluster and leave the Extension Unit through a single Output Pin.

The Pin channels on each Input Pin i of the Extension Unit are numbered from one to N_i. The PCC value N_i for each of the Input Pins is inherited from the first upstream Entity that defines a PCC value on its Output Pin.

On each Input Pin i, the incoming Cluster logical channels, numbered from one to n_i, are mapped one-to-one onto the Pin channels on that Input Pin, i.e. for each Input Pin i, Cluster channel 1 is mapped onto Pin channel 1, Cluster channel 2 is mapped onto Pin channel 2, and so on, up to Cluster channel n_i which is mapped onto Pin channel n_i. Note that the number of Pin channels on each of the Extension Unit's Input Pins (N_i) may be different from the actual number of channels in the incoming Clusters (n_i). However, for each Input Pin i, $N_i \geq n_i$ shall always be true.

AudioControls residing on currently unused Pin channels shall always remain accessible and retain their last setting for as long as the Extension Unit remains in a Power State that requires this.

If the Bypass Control is present, then engaging the bypass function shall result in passing the incoming Cluster on Input Pin one (the dominant Input Pin) unaltered to the output of the Unit. If it is necessary to be able to bypass the Extension Unit's functionality but providing an output Cluster different from the input Cluster on Input Pin one, then an explicit bypass topology using a Selector Unit should be implemented.

An Extension Unit shall implement the Bypass Control so that a generic audio driver that does not understand what functionality is implemented in the Extension Unit will be capable of removing it from the signal path.

The Extension Unit can expose multiple Cluster Configurations on its Output Pin. The Cluster Control will typically be implemented as Read (r) and informs Host software which Cluster Configuration is currently active. In this case, the currently active Cluster is determined by the vendor-defined internal operation of the Extension Unit.

However, the Cluster Control may be implemented as Read-Write (rw) and may be used by Host software to select the desired Cluster Configuration on the Output Pin. In other words, the selection of the Cluster Configuration

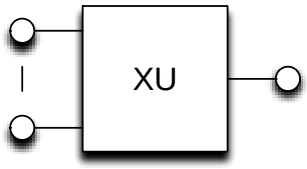
drives the vendor-defined internal operation of the Extension Unit based on the Cluster Configuration that the Host Software requires at the Output Pin of the Extension Unit. A Cluster Control value of zero indicates that the output Cluster is inherited from the current Cluster on Input Pin one (the dominant Input Pin). The Read-Only Cluster Active Control Shall always be implemented and indicates whether the output Cluster is currently Active or not.

Since the Extension Unit likely redefines the channels in its output Cluster as a processed result of any or all the incoming channels, it would be appropriate to treat these channels as independent of all other channels in the Audio Function and therefore assign them unique IDs in their respective **wChannelID** fields in the output Cluster.

If the Extension Unit is an explicit member of a Power Domain, then switching the Power Domain to any Power State other than PS0 or PS1 shall render the Unit non-functional, and its output Cluster shall be Inactive.

The symbol for the Extension Unit can be found in the following figure:

Figure 3-22: Extension Unit Icon



3.13.12 CLOCK ENTITIES

Clock Entities are special in the sense that they do not directly manipulate logical audio streams. Instead, they provide the functionality needed to manipulate sampling clock signals and clock routing for the different Input and Output Terminals and Sampling Rate Converter Units within the Audio Function. A Terminal inside the Audio Function shall only be connected to one Clock Entity. The clock signal present at the Clock Input Pin of a Terminal or SRC Unit determines the sampling frequency at which the underlying hardware is operating. If there is no need for the Audio Function to expose any clock information related to a Terminal or (part of) an SRC Unit, the Clock Input Pin of the Terminal or SRC Unit may be left unconnected.

3.13.12.1 CLOCK SOURCE

A Clock Source Entity provides a sampling clock signal on its single Clock Output Pin that is frequency-locked to the Reference Clock of the Clock Domain of which the Clock Source Entity is part. The actual output frequency of the Clock Source Entity can be different from the frequency of the Clock Domain's Reference Clock. Multiple Clock Source Entities can be part of the same Clock Domain, all sharing the same Reference Clock. Although they all may have a different output frequency, their output clock signals are all frequency-locked to one another since they are all derived from the same Reference Clock. By interacting with the Sampling Frequency Control inside the each Clock Source Entity, Host software can retrieve or alter the current output sampling frequency of the Clock Source Entity. Note that the Reference Clock frequency of the Clock Domain that is used by a Clock Source Entity is not exposed to the Host.

All different sampling clocks used inside the Audio Function shall be represented by separate Clock Source Entities. Even if the clock is generated 'inside a Terminal', that clock needs to be represented by a Clock Source Entity. As an example, a sampling clock could be recovered based on the amount of audio samples coming into the Audio Function over a USB OUT adaptive Endpoint. Alternatively, a sampling clock may be derived from the S/PDIF signal coming into the Audio Function on an external connector.

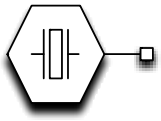
Note: In the case of an adaptive isochronous data Endpoint that support only a discrete number of sampling frequencies, the Endpoint shall at least tolerate ±1000 PPM inaccuracy on the reported Sampling Frequency Control values to accommodate sample clock inaccuracies.

The Clock Source Entity descriptor contains a field that indicates the Clock Domain of which the Clock Source Entity is part. Furthermore, since Input and Output Terminals only have a Clock Input Pin, a clock signal shall never be generated from a Terminal directly.

The output of a Clock Source Entity does not have to be always valid. For instance, if a Clock Source Entity represents an external sampling clock input on the Audio Function, the output of that Clock Source may not be valid when there is nothing connected to the external clock input. The Clock Source can always be queried for the validity of its output signal.

The symbol for the Clock Source Entity can be found in the following figure:

Figure 3-23: Clock Source Icon



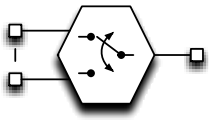
3.13.12.2 CLOCK SELECTOR

A Clock Selector Entity provides the functionality to select among different available sampling clock signals. It shall have two or more Clock Input Pins from which one is routed to the single Clock Output Pin.

Switching between Clock Inputs may be Host controlled (the Clock Selector's Selector Control is programmable via the appropriate Push Command) or the Audio Function may switch Clock Inputs due to some external event. A Clock Selector may support both control methods. The Selector Control can notify the Host of the change by generating an interrupt.

The symbol for the Clock Selector Entity can be found in the following figure:

Figure 3-24: Clock Selector Icon



3.13.13 CONNECTOR ENTITIES

Connector Entities are not explicitly exposed in the Audio Function topology. A Connector, or rather the signal(s) it carries is represented inside the Audio Function via the associated Input or Output Terminal. The Connector Entity provides optional AudioControls that advertise the insertion state of the Connector, and potentially also information about the nature and characteristics of the external device that is currently plugged into the Connector.

3.13.14 POWER DOMAINS AND POWER DOMAIN ENTITIES

Power Domains and their associated Power Domain Entities are not explicitly exposed in the Audio Function topology. Rather, a Power Domain bundles Entities whose Power States are managed together. The associated

Power Domain Entity provides AudioControls to impact the power behavior of the Power Domain through Commands, addressed to the Power Domain Entity.

3.13.15 CHANNELGROUPS, ENTITYGROUPS, AND COMMITGROUPS

ChannelGroups, EntityGroups and CommitGroups are not explicitly exposed in the Audio Function topology. A ChannelGroup shall have only channels as its Members. An EntityGroup shall have only Entities as its Members. A CommitGroup shall have only AudioControls as its Members.

3.14 OPERATIONAL MODEL

A Device may support multiple configurations. Within each configuration there may be multiple interfaces, each possibly having Alternate Settings. These interfaces may pertain to different functions that co-reside in the same composite Device. Even several independent Audio Functions may exist in the same Device. Interfaces belonging to the same Audio Function are grouped into an Audio Interface Association (AIA). If the Device contains multiple independent Audio Functions, there shall be multiple Audio Interface Associations, each providing full access to their associated Audio Function.

As an example of a composite Device, consider a computer display equipped with a built-in stereo speaker system. Such a Device could be configured to have one interface dealing with configuration and control of the monitor part of the Device (HID Class), while an Association of two other interfaces deals with its audio aspects. One of those, the AudioControl Interface, is used to control the inner workings of the function (Control etc.) whereas the other, the AudioStreaming Interface, handles the data traffic, sent to the monitor's audio subsystem.

The AudioStreaming Interface could be configured to operate in mono mode (Alternate Setting *x*) in which only a single channel data stream is sent to the Audio Function. The receiving Input Terminal would then output a mono cluster that could feed into an Up/Down-mix Processing Unit that duplicates this mono audio stream into two logical channels at its Output Pin, and those could then be reproduced on both speakers. From an interface point of view, such a setup requires one isochronous Endpoint in Alternate Setting *x* of the AudioStreaming Interface to receive the mono audio data stream, in addition to the mandatory control Endpoint and optional interrupt Endpoint in the AudioControl Interface.

The same system could be used to play back stereo audio. In this case, the stereo AudioStreaming Interface is selected (Alternate Setting *y*). This Interface also consists of a single isochronous Endpoint, now receiving a data stream that interleaves Front Left and Front Right channel samples. The receiving Input Terminal then splits the stream into a Front Left and Front Right logical channel and outputs a stereo Cluster that feeds into the Up/Down-mix Processing Unit. Rather than duplicating the mono channel, the Processing Unit now simply passes the incoming stereo Cluster unaltered to its Output Pin. From an interface point of view, this setup requires one isochronous Endpoint in Alternate Setting *y* of the AudioStreaming Interface to receive the stereo audio data stream. The AudioControl Interface Alternate Setting remains unchanged.

If the above AudioStreaming Interface were an asynchronous sink, one extra isochronous Feedback Endpoint would also be necessary.

As stated earlier, Audio functionality is located at the interface level in the Device Class hierarchy. The following sections describe the Audio Interface Association, containing a single AudioControl Interface and optional AudioStreaming Interfaces, together with their associated Endpoints that are used for Audio Function control and for audio data stream transfer.

3.14.1 AUDIOCONTROL INTERFACE

To control the functional behavior of a particular Audio Function, the Host manipulates the Entities inside the Audio Function. To make these objects accessible, the Audio Function shall expose a single AudioControl Interface. This Interface may contain the following Endpoints:

- A control Endpoint for manipulating Entity Control settings and retrieving the state of the Audio Function. This Endpoint is mandatory, and the default Endpoint 0 is used for this purpose.
- An interrupt Endpoint. The Endpoint is optional but shall be implemented if any of the AudioControls inside the Device have the need to generate and interrupt to notify the Host of a change in the Audio Function's behavior.

The AudioControl Interface is the only entry point to access the internals of the Audio Function. All Commands that are concerned with the manipulation of AudioControls within the Audio Function's Entities shall be directed to the AudioControl Interface of the Audio Function. Likewise, all Descriptors related to the internals of the Audio Function are part of the class-specific AudioControl Interface Descriptor.

The AudioControl Interface of an Audio Function shall only support a single Alternate Setting (Alternate Setting 0).

3.14.1.1 CONTROL ENDPOINT

The audio interface class uses Endpoint 0 (the default pipe) as the standard way to control the Audio Function using class-specific Commands. These Commands are always directed to one of the Entities that make up the Audio Function. The format and contents of these Commands are detailed further in this document.

3.14.1.2 INTERRUPT ENDPOINT

A USB AudioControl Interface may support an optional interrupt Endpoint to inform the Host about dynamic changes that occur on the different addressable Entities inside the Audio Function. The interrupt Endpoint is used by the entire Audio Interface Association to convey change information to the Host. It is considered part of the AudioControl Interface because this is the anchor interface for the AIA.

3.14.2 AUDIOSTREAMING INTERFACE

AudioStreaming Interfaces are used to interchange digital audio data streams between the Host and the Audio Function. They are optional. An Audio Function may have zero or more AudioStreaming Interfaces associated with it, each possibly carrying data of a different nature and format. Each AudioStreaming Interface shall have at most one isochronous data Endpoint. This construction guarantees a one-to-one relationship between the AudioStreaming Interface and the single audio data stream, related to the Endpoint. In some cases, the isochronous data Endpoint is accompanied by an associated isochronous explicit feedback Endpoint for synchronization purposes. The isochronous data Endpoint and its associated feedback Endpoint shall follow the Endpoint numbering scheme as set forth in the *USB Core Specifications*.

An AudioStreaming Interface may have Alternate Settings that can be used to change certain characteristics of the Interface and its underlying Endpoint. A typical use of Alternate Settings is to provide a way to change the subframe size and/or number of channels on an active AudioStreaming Interface. Whenever an AudioStreaming Interface requires an isochronous data Endpoint, it shall at least provide the default Alternate Setting (Alternate Setting 0) with zero bandwidth requirements (no isochronous data Endpoint defined) and one additional Alternate Setting that contains the actual isochronous data Endpoint. All non-zero Alternate Settings of an AudioStreaming Interface shall use the same data Endpoint and explicit feedback Endpoint, if present. More specifically, the data Endpoint shall use the same Endpoint number for all non-zero Alternate Settings of the Interface. Any Alternate

Setting which has an explicit feedback Endpoint shall use the same Endpoint number in all non-zero Alternate Settings.

The class-specific AudioStreaming Interface Descriptor contains two fields (**wStartDelayUnits** and **wStartDelay** field) that together indicate how much time it takes this Interface to reliably produce valid outgoing data (i.e., valid audio sample data and valid packet sizes – see Section 7.2.1.2.1, "Service Interval Packet Size Calculation") or effectively consume incoming data. This time is measured using as a reference the first IN or OUT PID that occurs at the start of an audio stream.

For proper streaming operation, it is highly recommended that the Host ignores all samples received between issuing the first IN PID and the expiration of the start delay time (as indicated by the **wStartDelay** field value) on input. Likewise, on output, the Host should send silence for at least the indicated start delay time after issuing the first OUT PID to avoid any undesired artifacts.

Switching from one active Alternate Setting to another active Alternate Setting on an AudioStreaming Interface shall never be performed directly. Instead, the Host software needs to first switch the Interface to its (mandatory) inactive Alternate Setting 0 and then, in a second step, switch the AudioStreaming Interface to the newly desired active Alternate Setting. The Device shall generate a Request Error on any standard SET_INTERFACE request that attempts to switch the Interface from one active Alternate Setting to a different active Alternate Setting.

For every defined AudioStreaming interface, there shall be a corresponding Input or Output Terminal defined in the Audio Function. For the Host to fully understand the nature and behavior of the connection, it needs to consider the Interface- and Endpoint-related Descriptors as well as the Terminal-related Descriptor.

Note: Interfaces and Endpoints are *USB Core Specification* concepts and therefore use a Host-centric terminology. Terminals are Audio Device Class concepts and therefore use Audio Function-centric terminology. *OUT* Interfaces and Endpoints correspond to *Input* Terminals, and *IN* Interfaces and Endpoints correspond to *Output* Terminals.

3.14.2.1 ISOCHRONOUS AUDIO DATA STREAM ENDPOINT

In general, the data streams that are handled by an isochronous audio data Endpoint do not necessarily map directly to the logical channels that exist within the Audio Function. As an example, consider the case where multiple logical audio channels are compressed into a single data stream (AC-3, WMA …). The format of such a data stream may be entirely different from the original format of the logical channels (for example, 640 kbits/s AC-3 5.1 audio as opposed to 6-channel 16-bit 44.1 kHz audio). Therefore, to describe the data transfer at the Endpoint level correctly, the notion of logical channel is replaced by the notion of audio data stream. It is the responsibility of the AudioStreaming Interface which contains the OUT Endpoint to convert between the audio data stream and the embedded logical channels before handing the data over to the Input Terminal. In many cases, this conversion process involves some form of decoding. Likewise, the AudioStreaming Interface which contains the IN Endpoint shall convert logical channels from the Output Terminal into an audio data stream, often using some form of encoding.

Requests to control properties that exist within an Audio Function, such as or Mute cannot be sent to the Endpoint in an AudioStreaming Interface. An AudioStreaming Interface operates on audio data streams and is unaware of the number of logical channels it eventually serves. Instead, these Commands shall be directed to the proper Audio Function Entities via the AudioControl Interface.

As already mentioned, an AudioStreaming Interface may have zero or one isochronous audio data Endpoint. If multiple synchronous audio channels need to be communicated between Host and Audio Function, they shall be

clustered into one physical Cluster by interleaving the individual audio data, and the result can be directed to the single Endpoint.

If an Audio Function needs more than one Cluster to operate, each Cluster is directed to the Endpoint of a separate AudioStreaming Interface, belonging to the same Audio Interface Association (all servicing the same Audio Function).

3.14.2.2 ISOCHRONOUS FEEDBACK ENDPOINT

For adaptive audio source Endpoints and asynchronous audio sink Endpoints, an explicit synchronization mechanism is needed to maintain synchronization during transfers. For details about synchronization at different Endpoint speeds, see the applicable *USB Core Specifications*.

3.14.2.3 AUDIO DATA FORMAT

The format used to transport audio data over the USB is entirely determined by the value in the **wFormat** field of the class-specific AudioStreaming Self Descriptor. Some additional fields in this Descriptor further describe the format. For details about the defined Format Types and associated data formats, see Section 7, “Audio Data Formats.”

3.14.3 AUDIOCONTROLS

Inside an Entity, functionality is described through AudioControls. Each AudioControl provides access to a specific audio characteristic, such as , Bass, etc.

Each Entity shall only contain a specific set of AudioControls, permitted for use by that Entity as defined by this specification.

It is important to note that all interactions with Entities in the Audio Function are performed via AudioControls. All accessible parameters inside these Entities are modeled using the concept of the AudioControl and its associated Attributes.

AudioControls are either associated with a particular Pin channel inside an Entity or influence the behavior of the Entity as a whole.

Each AudioControl has a set of Attributes that can be manipulated or that present information about the behavior of the AudioControl. An AudioControl has the following Attributes:

- Current Attribute (CUR) – Mandatory.
- Next Attribute (NEXT) – Optional.
- Capabilities Attribute (CAP) – Mandatory.
- Range Attribute (RANGE) – Optional.

Attributes may be implemented as Read[-Only] (r), Read-Write (rw) and even Write[-Only] (w).

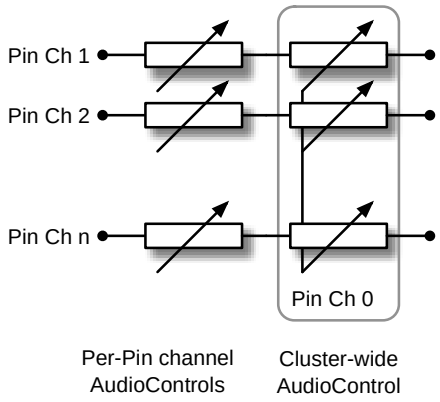
An Attribute of an AudioControl provides the finest level of addressable control granularity within the Audio Function. See Section 5.3.2, “AudioControl Commands” for information on how to access a particular AudioControl.

3.14.3.1 CLUSTER-WIDE AUDIOCONTROLS

AudioControls are often implemented on a per-Pin-channel basis. However, to control the settings of an AudioControl as a whole, the concept of a Cluster-wide AudioControl has been defined. Changing the value of a

Cluster-wide AudioControl applies that change to all Pin channels simultaneously. Note that a Cluster-wide AudioControl shall always be implemented separate from and independent of the per-Pin-channel AudioControls. Changing the setting of a Cluster-wide AudioControl shall not affect the settings of any of the individual Pin channel AudioControl settings. The following figure illustrates the concept.

Figure 3-25: N-channel AudioControl with Cluster-wide AudioControl



3.14.3.2 SYNCHRONIZING MULTIPLE AUDIOCONTROLS

In some cases, multiple AudioControls need to be manipulated simultaneously to achieve a desired effect or to avoid unwanted transitory side effects. To accomplish this, every AudioControl may have a Next Attribute that is used to preload a certain value for the AudioControl. Only when a Commit Command is issued (to the Audio Function as a whole or to a CommitGroup) will all AudioControls in the Audio Function or those in the CommitGroup take on as their Current Attribute setting the value that was preloaded in their Next Attribute. This effectively provides a means to have several AudioControls (including those present in the AudioStreaming Interfaces and all other Entities) take on new values simultaneously. Note further that any AudioControls in the entire Audio Function may be synchronized using this technique by properly interleaving sets of Push(NEXT) Commands with Commit Command sequences.

For more detailed information on AudioControl and Attribute manipulation, see Section 5, “Commands & Requests.”

3.14.4 CLOCK MODEL

Clock Entities provide a way to accurately describe the use and distribution of sampling clock signals throughout the Audio Function. Sampling frequencies inside the Audio Function may only be influenced by directly interacting with the Sampling Frequency Control inside a Clock Source Entity. The Sampling Frequency Control RANGE Attribute provides the necessary information for Host software to determine what sampling frequencies the AudioControl (and thus the Clock Source Entity) supports.

A side effect of changing the sampling frequency could be that certain AudioStreaming Interfaces may need to switch to a different Alternate Setting to support the bandwidth needed for the new sampling frequency. This specification does not allow an AudioStreaming Interface to switch from one Alternate Setting to another on its own except to change to Alternate Setting zero, which is the idle setting. Instead, when the Audio Function detects that it can no longer support a certain Alternate Setting on an AudioStreaming Interface, it shall switch to Alternate Setting zero on that Interface and report the change to Host software through the Active Alternate Setting Control

interrupt. The Host can then query the Interface for new valid Alternate Settings for the Interface through the Get Valid Alternate Settings Control Command and make an appropriate selection.

Note: To keep the number of Alternate Settings in an AudioStreaming Interface to a minimum, it is not recommended to provide a separate Alternate Setting for every supported sampling frequency. A few Active Alternate Settings (low bandwidth, medium bandwidth, high bandwidth) may be enough to provide reasonable bandwidth control.

Audio streams can be bridged from one Clock Domain to another using the Sampling Rate Converter Unit.

3.14.5 CONNECTOR MODEL

Each Connector has its own Connector Entity and is identified by a unique Connector ID. The Connector Entity optionally has an embedded Insertion Detect Control that indicates through its CUR Attribute whether there is currently an external device plugged into the Connector that carries information related to the Terminal to which the Connector is associated.

It is important to note that a Connector and its associated Connector Entity in the context of this specification may represent only part of a physical external connector on the Device. For example, an external physical connector that can accept both headphones and headsets would be represented by two distinct Connectors and Connector Entity pairs, where one Connector and its Connector Entity would be associated with an Input Terminal (for the microphone part) and another Connector and its Connector Entity would be associated with an Output Terminal (for the speaker part). Each corresponding Connector Entity could have its own independent Insertion Detect Control. One would detect the insertion of the microphone part of a headset, whereas the other would detect the insertion of the speaker part of either a headphone or a headset.

The Insertion Detect Control change generates an interrupt whenever the Audio Function autonomously detects a change in insertion state.

A Connector Entity is always associated with one or more Input or Output Terminals through which the signals, carried over the Connector enter or leave the Audio Function. Multiple Connector Entities may be associated with the same Terminal. This indicates that there is a functional relationship among the Connectors, and they should be considered conceptually as a whole but with an implementation that requires multiple physical plugs or receptacles at the same time. A typical example of this is a set of three 3.5 mm receptacles that are used to connect a 5.1 surround-capable speaker set to the Audio Device. One Connector carries Front Left/Right signals, the second carries Surround Left/Right signals, while the third one carries Center/LFE signals typically.

3.14.6 POWER DOMAIN MODEL

Managed Power Domain support is optional for an Audio Function. The Audio Function has zero or more Power Domains, each represented by a Power Domain Entity that is identified by a unique Power Domain ID. Each Power Domain can be individually manipulated by the Host to optimize overall power consumption. If implemented, the Audio Function contains a set of Power Domain Entities with their embedded AudioControls that allows the Host to set Power States for each individual Power Domain inside the Audio Function. The Audio Function contains as many Power Domain Entities as there are Power Domains and access to a specific Power Domain Control is based on the Power Domain Entity ID. A Power Domain contains one or more member Entities. Power Domain Entities shall never be a member of a Power Domain. An Entity shall not be a member of more than one Power Domain.

A Power Domain shall support five Power States PS0 to PS4.

Power State PS0 is the fully operational state and shall be the default Power State for all Power Domains.

Power State PS1 is a functional state where power consumption shall be no higher than in Power State PS0. Audio fidelity may be reduced in this Power State. All AudioControls and bypass functionality residing in Entities that are members of the Power Domain shall remain operational and memory content such as downloaded algorithms shall be preserved.

Power State PS2 is a state where audio streaming over USB Endpoints shall not occur, and power consumption shall be no higher than in Power State PS1. All AudioControls and bypass functionality residing in Entities that are members of the Power Domain shall remain operational and memory content such as downloaded algorithms or buffered audio shall be preserved.

Power State PS3 is a state where audio streaming on the Output Pins of the member Entities shall not occur unless the optional Bypass functionality is engaged. Power consumption shall be no higher than power consumption in Power State PS2. All AudioControls and bypass functionality residing in member Entities shall remain operational. Memory content is not guaranteed to be preserved. Any Entity that is affected by the loss of memory may revert to unspecified values for its AudioControls when transitioning to a higher Power State.

Power State PS4 is a non-functional state where audio streaming on the Output Pins of the member Entities shall not occur, and power consumption shall be no higher than power consumption in Power State PS3. AudioControls and bypass functionality residing in member Entities shall become non-operational and may lose state. Memory content is likely to be lost. Any Entity that is affected by the loss of memory may revert to unspecified values for its AudioControls when transitioning to a higher Power State. This allows the Audio Function to completely remove power from the Power Domain. Note that Power Domain Entities themselves shall never be part of a Power Domain and their AudioControls shall always be operational.

Note: The Audio Function shall ignore any traffic on its USB streaming Endpoints when their associated Terminals are in Power States PS2 or below.

An Audio Function is not allowed to change the Power State of any of its Power Domains autonomously. A change in Power State shall always be initiated through an explicit Command from the Host. Power States shall be retained across Function or Device suspend. All transitions from any Power State PSx to any other Power State PSy are allowed.

Note: Power States operate independently from the Function or Device suspend state. For example, bringing all Power Domains in an Audio Function into Power State PS4 does not automatically put the USB Device into USB suspend.

An Audio Function shall always honor a Command to change the Power State of a Power Domain.

Actual power consumption levels for each Power State are not advertised. The only requirement is that a higher numbered Power State consume no more power than all lower numbered Power States, potentially at the expense of higher exit times. Entry time is defined as the approximate time it takes to get from the fully operational Power State PS0 to a lower Power State. Exit time is defined as the approximate time it takes to get from a lower Power State back to the fully operational Power State PS0. The Power Domain Entity Descriptor indicates the entry and exit times for Power States PS1 to PS4. Intermediate entry and exit times (from PSn to PSm where both n and m are non-zero) are not listed but shall never be larger than the corresponding PS0 to PSn times (for entry times) or PSn to PS0 times (for exit times).

The Audio Function is best placed to manage the details of its resources and their power consumption under various conditions. These details are therefore not exposed to the Host. Rather, the Host manages the Power States in each of the Power Domains to indicate to the Audio Function which parts of the Audio Function it intends to use.

The Power State Control shall be used by Host software to Command a Power State change or return the current Power State.

3.14.6.1 INTERDEPENDENCY AMONG POWER STATES, BYPASS FUNCTIONALITY AND OUTPUT CLUSTERS

The Cluster present on the Output Pin of an Entity depends, among other things, on the Power State of the Entity, the state of the Bypass Control, and the state of the Cluster Control.

The following table summarizes the required behavior.

Table 3-1: Output Cluster Configuration and Cluster Content Behavior

<table>
<tr><th colspan="2">Bypass Control</th><th colspan="2">Not Present or Disabled</th><th>Enabled</th></tr>
<tr><th colspan="2">Cluster Control</th><th>n > 0</th><th>0</th><th>Don’t Care</th></tr>
<tr><td rowspan="5">Power State Ctrl</td><td>PS0</td><td rowspan="3">Cluster Configuration set to n.
Cluster Content generated internally</td><td rowspan="3">Cluster Configuration inherited from Input Pin 1.
Cluster Content generated internally</td><td rowspan="3">Cluster Configuration and Cluster Content inherited from Input Pin 1</td></tr>
<tr><td>PS1</td></tr>
<tr><td>PS2</td></tr>
<tr><td>PS3</td><td rowspan="2">Inactive</td><td rowspan="2">Inactive</td></tr>
<tr><td>PS4</td><td>Inactive</td></tr>
</table>

3.14.7 ADDITIONAL POWER CONSIDERATIONS AND REQUIREMENTS

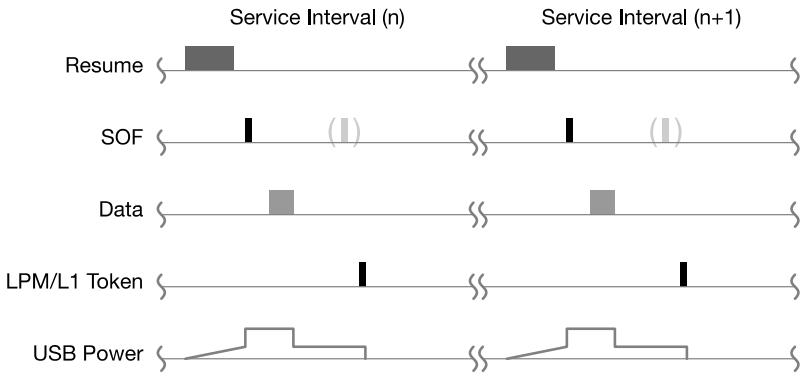
It is strongly recommended that an Audio 4.0 compliant High Speed Device support LPM/L1. When supported, the implementation shall follow the requirements defined in the *USB 2.0 LPM/L1 ECN* and *LPM Errata*. LPM/L1 is not applicable to Full Speed or SuperSpeed Devices. LPM/L1 enables both the Device and Host to save USB subsystem power by entering “sleep” during L1 power state. Sleep power is typically similar to Suspend power.

3.14.7.1 USB AUDIO 4.0 LPM/L1 IMPLEMENTATION EXAMPLE

Isochronous transfers for legacy USB audio (UAC1 or UAC2) typically occur every 1 ms for Full Speed USB. The bus is busy during a large part of the service interval. Experience has shown that legacy USB audio cannot replace the 3.5 mm phone jack in mobile phones and other portable, battery powered products, as power consumption is too high during audio playback use case. An alternative implementation is High Speed USB and UAC2, where audio is transferred in short bursts. There are some power savings due to the bus being idle during most of the service interval. However, the bus cannot enter Suspend L2 for further power savings since L2 entry/exit latency is ~10 ms or more.

The USB 2.0 Link Power Management (LPM) specification defines a new L1 power state (LPM/L1) for High Speed USB, with directed L1 entry and short L1 exit latency. The combination of High Speed USB bursting and the L1 power state offers significant power savings opportunities for USB Audio 4.0 devices and hosts. The figure below shows typical events for an LPM/L1 capable USB 4.0 Audio device.

Figure 3-26: Typical behavior of Audio Device that implements LPM/L1



Note: There is repeated LPM/L1 exit (Resume), data burst, and LPM/L1 entry sequences. Actual timing is Host and implementation dependent. Typical Host implementations enable L1 power state ~50 % of the time with a 1 ms Service Interval and 85-90 % with a 4 ms Service Interval. Note that SOF(s) after the data burst are optional and Host implementation dependent.

3.14.7.2 LPM/L1 BACKGROUND INFO

From a USB point of view, LPM/L1 is directed Suspend, derived from SuperSpeed U1/U2 power states. LPM/L1 fast entry uses an extended LPM Token, while LPM/L1 fast exit uses short duration Host Initiated Resume Signaling. Active data transfer times for typical audio streams are typically less than 20 µs per each ms of Service Interval. Even considering LPM/L1 entry and exit overhead, significant power savings are possible with 1 ms Service Interval. Audio 4.0 Devices may offer 2 ms or 4 ms Service Intervals for even more power savings. This is especially useful when Audio 4.0 Devices are used with battery powered Hosts supporting audio-only playback use cases.

3.14.7.3 DEVICE LPM/L1 IMPLEMENTATION REQUIREMENTS

An LPM/L1 capable Audio 4.0 Device shall present support for LPM/L1 in the extended USB BOS descriptor. The Device shall present a descriptor corresponding to 75 µs or 100 µs Host Initiated Resume Delay (HIRD). An Audio 4.0 Device shall accept the LPM/L1 entry Commands (i.e., return an ACK), even if the Device's isochronous RX Endpoint buffer has not yet been emptied or the isochronous TX Endpoint already contains samples for the next audio burst. During the L1 power state, the Audio 4.0 Device USB subsystem shall enter the L1 low-power mode aka 'sleep' with lower power consumption than during 'idle' i.e., no active data transfers.

3.14.7.4 HOST LPM/L1 BEHAVIOR NOTES

A USB Host that supports LPM/L1 is required to follow the requirements defined in the USB 2.0 LPM/L1 ECN and LPM Errata. The Host may decide to not use LPM/L1 even if the Audio 4.0 Device supports LPM/L1. When LPM/L1 is enabled, LPM/L1 capable Hosts may, and typically will (but are not required to) Command LPM/L1 entry after servicing the Audio 4.0 Device Endpoint(s) in each Service Interval. When in L1, the Host is responsible for, and can be trusted to resume the Device in time to service the next Service Interval.

3.14.7.5 USB AND AUDIO SYNCHRONIZATION

For isochronous Endpoints, the Host will send one or more SOF tokens before resuming transactions to the Device Endpoint(s). The Host may (but is not required to) send one or more SOF tokens before Commanding LPM/L1 entry

after servicing the Endpoint(s). As indicated in Figure 3-26, a highly optimized host can send an LPM/L1 Token immediately after the Data burst.

An LPM/L1 capable Audio 4.0 Device can use sequential SOF tokens to synchronize its internal clock with the USB clock. If the Host does not send enough consecutive SOF tokens, and/or the Device needs additional SOF tokens in order to resynchronize its internal clock with the USB clock, the Device may NAK the Command to put the link into an L1 state after the Host has serviced the endpoint(s) in the current Service Interval; this ensures that the link stays in L0 until the next Service Interval. The Device shall NAK the LPM/L1 token no more than once every 64 ms to ensure that LPM/L1 usage does save power.

3.14.8 GROUP MODEL

Group support is optional for an Audio Function. This specification defines three group types as indicated in Section 3.10, “Groups.”

3.14.8.1 CHANNELGROUP

ChannelGroups are an integral part of a Cluster and groups together closely related audio channels in the Cluster. The ChannelGroup is a lightweight construct, consisting of a list of Channel IDs of the related channels.

3.14.8.2 ENTITYGROUP

EntityGroups group together Entities that have a relationship to or are associated with one another in some form. For example, some Input Terminals that represent the various microphones of a sophisticated gaming headset, and an Output Terminal that represents the earpieces (headphone part) of that headset can be grouped together to indicate that they are part of the same device. The EntityGroup is a lightweight construct, consisting of an EntityGroup Descriptor that simply enumerates the Members of the EntityGroup by listing their respective Entity IDs. An EntityGroup does not have any associated AudioControls. Connector Entities are typically not included as Members of an EntityGroup. The Connectors associated with a Terminal are referenced directly from within the Terminal Descriptor.

3.14.8.3 COMMITGROUPS

CommitGroups allow the Audio Function to indicate to Host software which AudioControls are best updated using the Commit Capability to avoid unwanted artifacts or provide the best user experience. As a generic example, multiple AudioControls that impact a single feature, such as those exposed by Effect Units or Processing Units, are best set to their desired values at the same point in time. In most cases, it would be even desirable to synchronously apply the desired changes across multiple channels as well.

The CommitGroup is a lightweight construct, exposed within the Audio Function through its CommitGroup Descriptor. A CommitGroup Descriptor lists all AudioControls that are best manipulated in a synchronous fashion to achieve smooth and artifact-free operation of a certain feature, effect or process. If supported, the Descriptor ID may be used in the Commit Command to only affect the AudioControls in that CommitGroup.

All AudioControls that are Members of a CommitGroup shall support the NEXT Attribute. Multiple CommitGroups may be exposed within the Audio Function. AudioControls may be members of multiple CommitGroups.

3.14.9 BINDING BETWEEN BUTTONS AND AUDIOCONTROLS

A Device that contains an Audio Function may also have one or more buttons/sliders/knobs that are intended to control certain aspects of the Audio Function inside the Device. An example is a Gain Control on a multimedia

speaker. Since an Audio Function may potentially contain many AudioControls of the same type, there is a need to bind a physical control (button, knob, slider, jog, …) to a particular AudioControl inside the Audio Function.

This specification provides two mutually exclusive methods to create this binding:

- The button is implemented using the Human Interface Device Class (*Universal Serial Bus Device Class Definition for Human Interface Devices (HID)*). In this case, it is the responsibility of the Host to establish the binding between the HID event and the action issued to the Audio Function(s)
- The button is an integral part of the AudioControl. In this case, the AudioControl shall notify the Host of any change through the interrupt mechanism.

It is prohibited to implement both methods for the same physical button. However, it is allowed to use the first method for some of the buttons and the second method for the remaining buttons. It is strongly discouraged to implement buttons that use neither of the above-mentioned methods, i.e., buttons that are invisible to Host software and have a local effect only.

3.14.9.1 BUTTON IS A HID CONTROL

In this case, the button is separate from the Audio Function and is implemented as part of a HID interface within the Device. Any change of state for the button is communicated to Host software via HID reports. It is then up to Host software to interpret the button state change and determine the appropriate action to be taken toward the Audio Function. Therefore, the binding responsibility resides entirely within the application or Operating System software. Although this method provides extensive flexibility, it also puts the burden of providing the correct binding on the software, making it sometimes hard to create generic application or OS software that generates the proper (manufacturer intended) binding.

3.14.9.2 BUTTON IS AN INTEGRAL PART OF THE AUDIOCONTROL

In this case, the button directly interacts with the actual AudioControl. The change of state of the AudioControl resulting from the button manipulation is reported to Host software through the AudioControl interrupt mechanism. Consequently, the binding between the button and the AudioControl is very direct and entirely dictated by the design of the Device. This method provides a very clear and straightforward binding between the button and the behavior of the Audio Function but prevents the Host from allocating the button for other purposes.

4 DESCRIPTORS

The following sections describe the standard and class-specific USB Descriptors for the Audio Interface Class.

4.1 STANDARD DESCRIPTORS

Because Audio functionality is always residing at the interface level, all relevant fields in the standard Descriptors shall indicate that class information is to be found at the interface level so that enumeration software looks down at the interface level to determine the Interface Class and to also ensure that IAD-aware enumeration software gets loaded.

4.2 CLASS-SPECIFIC DESCRIPTORS

Note: In this 4.0 version of the Specification, High Capability Descriptors as defined in version 3.0 of the Audio Device Class-specification are renamed to Extended Descriptors. Likewise, the Get High Capability Descriptor Command is renamed to Get Extended Descriptor Command.

This specification uses Extended Descriptors to express most of its class-specific Descriptors. Extended Descriptors are never part of the configuration Descriptor hierarchy, returned by the Get Configuration Command. Only traditional layout Descriptors shall be included in this hierarchy. An Extended Descriptor shall therefore always be referenced by a traditional class-specific Descriptor that includes the Extended Descriptor's unique ID as one of its fields.

In the remainder of this specification, the "Extended" designation may be omitted when referring to a class-specific Extended Descriptor. Furthermore, the designation "class-specific" may be omitted when the context makes it obvious that a class-specific Descriptor is referenced.

4.2.1 TRADITIONAL CLASS-SPECIFIC DESCRIPTOR CHARACTERISTICS

Traditional class-specific Descriptors as defined in this specification all follow a common layout. The first three fields of any traditional class-specific Descriptor are common to all traditional Descriptors and are followed by a layout that is specific to the type and subtype of the descriptor.

The **bLength** field contains the total length of the Descriptor, in bytes.

The **bDescriptorType** field in part follows the bit allocation scheme of the **bmRequestType** field and identifies the Descriptor as being a class-specific Descriptor. Bit D7 of this field is reserved and shall be set to zero. Bits D6..5 are used to indicate that this is a class-specific Descriptor (D6..5 = 0b01). Bits D4..0 are used to encode the Descriptor type.

The **bDescriptorSubtype** field further qualifies the exact nature of the Descriptor.

Table 4-1: Traditional Class-specific Descriptor Layout

Offset	Field	Size	Value	Description
0	bLength	1	Number	Size of this Descriptor, in bytes: 3+n.
1	bDescriptorType	1	0b001xxxxx	Descriptor type.
2	bDescriptorSubtype	1	Constant	Descriptor subtype.
3	---	n	---	Descriptor type- and subtype-specific Descriptor layout.

4.2.2 EXTENDED CLASS-SPECIFIC DESCRIPTOR CHARACTERISTICS

Most class-specific Descriptors of an Audio Function are not retrieved during enumeration time. Rather, they can only be retrieved using the Get Extended Descriptor Command as defined in Section 5.3.5.2, “Extended Descriptor ”.

Note: The only class-specific Descriptors that are retrieved during enumeration are the traditional class-specific Descriptors of Type CS_INTERFACE and Subtype AC_GENERIC or AS_GENERIC.

An Extended Descriptor is always referenced by another Descriptor via its Audio Function-wide unique ID.

An Extended Descriptor may be up to 64 KiB in length. Also, it shall generate an interrupt of source type EXTENDED_DESCRIPTOR whenever the Audio Function changes the Descriptor during normal operation (dynamic Descriptor).

Extended Descriptors all follow a common layout. The first five fields of any Extended Descriptor are common to all Extended Descriptors and are followed by a layout that is specific to the type and subtype of the Descriptor. The common fields are:

- The **wLength** field contains the total length of the Descriptor, in bytes. Descriptor lengths up to a maximum of 65,535 bytes are supported.
- The **wDescriptorType** field in part follows the bit allocation scheme of the **bmRequestType** field as defined by the standard USB Request and identifies the Descriptor as being a class-specific Descriptor. Bits D15..7 of this field are reserved. Bits D6..5 are used to indicate that this is a class-specific Descriptor (D6..5 = 0b01). Bits D4..0 are used to encode the Descriptor type.
- The **wDescriptorSubtype** field further qualifies the exact nature of the Descriptor.
- The **wDescriptorID** field contains a value that uniquely identifies the Extended Descriptor within the Audio Function. The value zero is reserved and shall not be used as a valid Descriptor ID.
- The **wStrDescriptorID** field contains the ID of a class-specific String Descriptor that provides additional descriptive information about the subject of this Descriptor. For example, a class-specific Clock Source Descriptor would use this field to provide a human-readable name for the Clock Source. This field shall be set to zero if there is no String Descriptor associated with the class-specific Extended Descriptor.

Table 4-2: Class-specific Descriptor Layout

Offset	Field	Size	Value	Description
0	wLength	2	Number	Size of this Descriptor, in bytes: 10+n.
2	wDescriptorType	2	Constant	Descriptor type.
4	wDescriptorSubtype	2	Constant	Descriptor subtype.
6	wDescriptorID	2	Number	Unique ID for this Descriptor.
8	wStrDescriptorID	2	Number	ID of a class-specific String Descriptor that provides additional information about the subject of this Descriptor.
10	---	n	---	Descriptor type- and subtype-specific Descriptor layout.

4.2.3 COMMON FIELDS IN SOME CLASS-SPECIFIC DESCRIPTORS

Some fields appear in more than one class-specific Descriptor. Instead of repeating the description of that field for all Descriptor instances, the description is indicated once in this Section and applies to all Descriptor instances in which the field appears.

4.2.3.1 wxxxID FIELD

Each Entity (Clock, Unit, Terminal, and Power Domain) within the Audio Function is assigned a unique identification number: The Clock Entity ID, Unit ID, Terminal ID, or Power Domain ID.

All Entity Descriptors have a common field at offset ten that contains the unique ID for the Entity. The value zero (0x0000) is used to identify the Interface itself.

Besides uniquely identifying all addressable Entities in an Audio Function, the IDs (except for the Power Domain ID) also serve to describe the topology of the Audio Function, i.e., the **wSourceID** field of a Unit or Terminal Descriptor indicates to which other Unit or Terminal this Unit or Terminal is connected. Likewise, the **wCSourceID** field in a Terminal or SRC Descriptor indicates to which Clock Entity this Terminal or SRC is connected. Furthermore, the Entity IDs are also used to indicate to which Power Domain each Entity belongs.

4.2.3.2 wxxxDescrID FIELD

Most class-specific Descriptors have one or more 2-byte fields with a name of the form “wxxxDescrID” that contains the ID of another class-specific Descriptor that either serves as an extension to the first Descriptor or refers to a String Descriptor that provides additional human-readable information. A value of zero in this field indicates that there is no extension Descriptor or String Descriptor associated with that first Descriptor.

4.2.3.3 dOptControls FIELD

The **dOptControls** field (or **daOptControls()** array elements) contains a bitmap, indicating which of the predefined optional AudioControls for that Entity are present. If a certain AudioControl is not present, then the bit shall be set to 0b0. If an AudioControl is present, then the bit shall be set to 0b1. Each implemented AudioControl provides all necessary information for the Host to determine how to interact with it via the mandatory Capability (CAP) Attribute. AudioControls that are mandated by this specification are *not* included in the **dOptControls** field.

4.3 AUDIO FUNCTION DESCRIPTOR SET

The standard interface association mechanism is used to describe an Audio Interface Association (AIA). All Interfaces belonging to the same AIA shall be identified by means of the standard Interface Association Descriptor (IAD).

Each AIA shall consist of the mandatory AudioControl Interface that shall be the first in the AIA (having the lowest interface number). All AudioStreaming Interfaces shall be contiguously numbered and immediately follow the AudioControl Interface in the AIA.

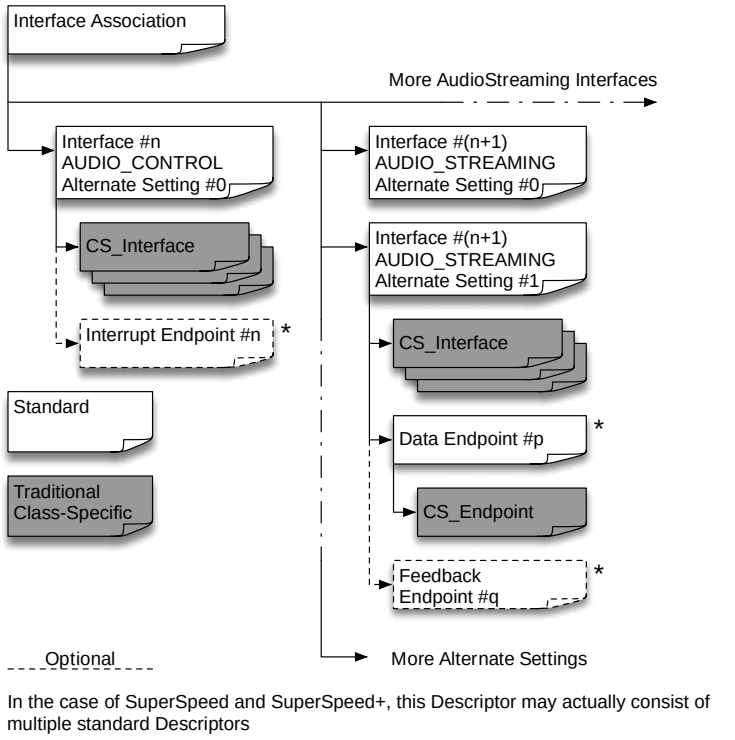
Note: For more information on Interface Association, refer to *USB Interface Association Descriptor Device Class Code and Use Model White Paper*, available on the USB web site.

The collection of the Interface Association Descriptor (IAD) and the full set of Standard Descriptors and class-specific Descriptors that together comprise the entire Audio Function is called the Audio Function Descriptor Set (AFDS).

4.3.1 AUDIO FUNCTION DESCRIPTOR SET LAYOUTS

In the Audio Device Class-specification Revision 2.0 (and lower), all Descriptors are of the traditional type as defined in the *USB Core Specifications*. All class-specific Descriptors are interspersed with the standard Descriptors to provide a full view of the Audio Function. The AFDS 2.0 is always retrieved from the Device at enumeration time as part of a Configuration Descriptor bundle. Note that in the Configuration Descriptor bundle, the IAD does not need to immediately precede the AudioControl Interface Descriptor to which the IAD refers. It is sufficient that the IAD is positioned earlier in the bundle than the AudioControl Interface Descriptor to which the IAD refers.

Figure 4-1: Audio 2.0 Function Descriptor Set layout

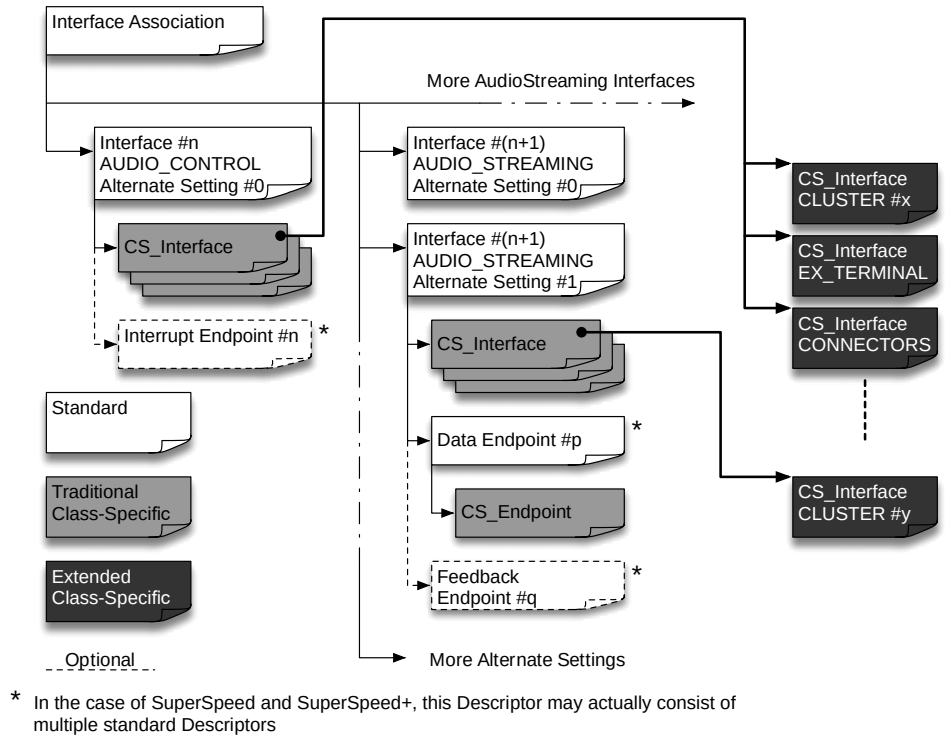


* In the case of SuperSpeed and SuperSpeed+, this Descriptor may actually consist of multiple standard Descriptors

Starting with Audio Device Class-specification Revision 3.0, the concept of the Extended Descriptor is introduced and some of the class-specific Descriptors are defined as Extended Descriptors.

Most of the class-specific Descriptors are still of the traditional type and interspersed with the standard Descriptors. The AFDS 3.0 is always retrieved from the Device at enumeration time as part of a Configuration Descriptor bundle. The Host then uses the Get Extended Descriptor Command to retrieve the Extended Descriptors whenever it encounters an Extended Descriptor ID in one of the traditional class-specific Descriptors. As such, the set of Extended Descriptors is not considered to be part of the AFDS.

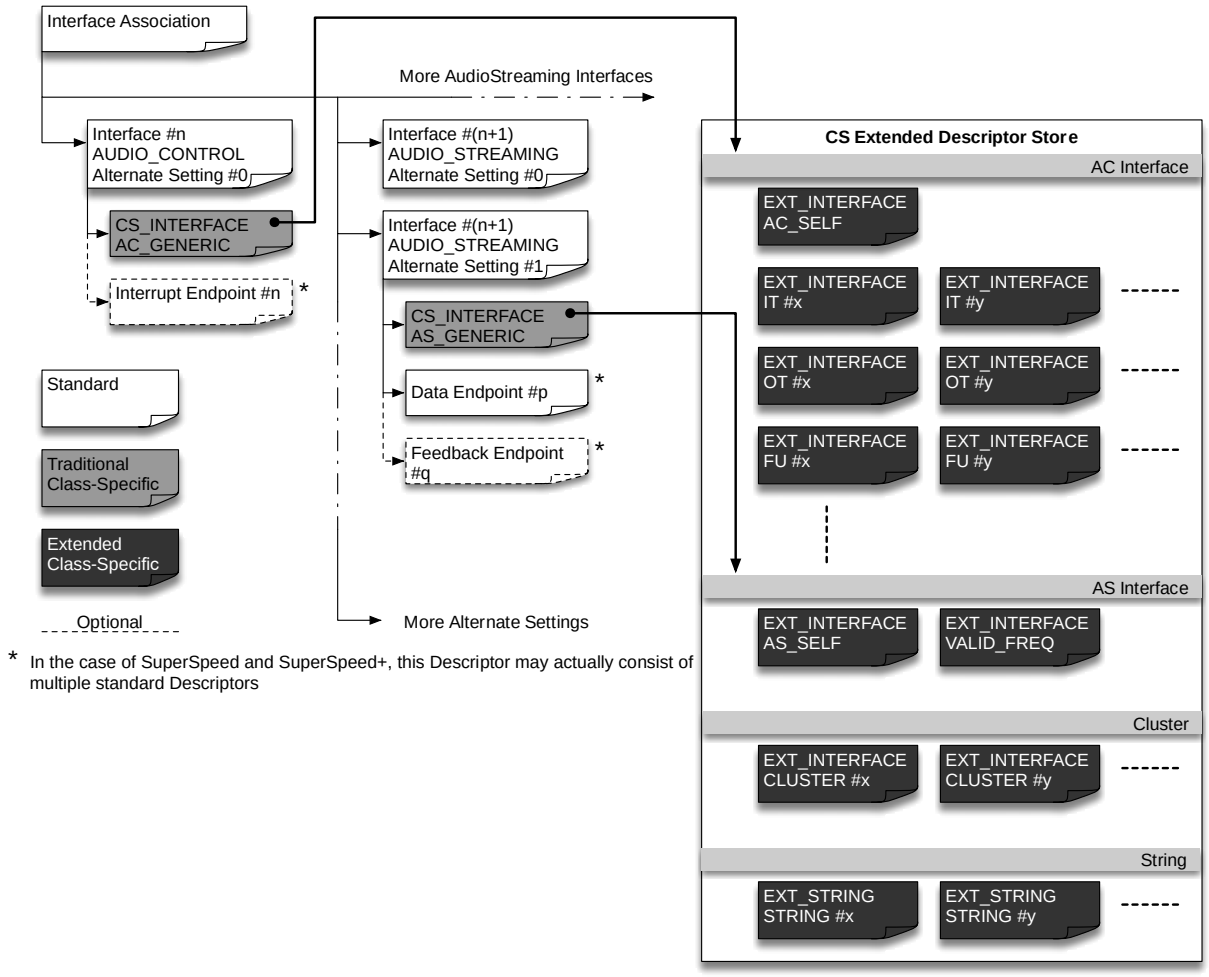
Figure 4-2: Audio 3.0 Function Descriptor Set layout



With this version 4.0 of the Specification, the use of Extended Descriptors has been further generalized. All class-specific information is now made available exclusively through Extended Descriptors as illustrated in Figure 4-3. The only traditional class-specific Descriptors that are interspersed with the standard Descriptors are Descriptors of type xx_GENERIC. The AFDS 4.0 is retrieved from the Device at enumeration time as part of a Configuration Descriptor bundle. Alternatively, an AFDS can be retrieved from the Device as a Higher Revision Level AFDS through other means as described in Section 8, “Backwards Compatibility Considerations”.

The class-specific Descriptors of type xx_GENERIC simply contain a list of Descriptor IDs. Each ID references an Extended Descriptor in the Extended Descriptor Store. The Host should consult the Descriptor Store to retrieve all class-specific information regarding the Audio Function. It uses the Get Extended Descriptor Command to retrieve each individual Descriptor in the Store. The Extended Descriptor Store is not considered to be part of the AFDS.

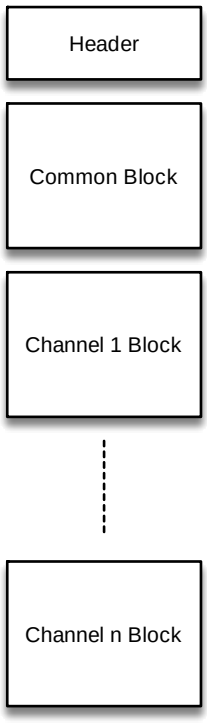
Figure 4-3: Audio 4.0 Function Descriptor Set layout



4.4 CLUSTER DESCRIPTOR

A Cluster is a grouping of audio channels that share the same characteristics such as sampling frequency, bit resolution, etc. The Cluster Descriptor is used to characterize the Cluster. At the highest level, it is structured as follows:

Figure 4-4: Cluster Descriptor



The Cluster Descriptor consists of a fixed Header, followed by an optional Common Block, followed by as many Channel Blocks as there are channels in the Cluster.

4.4.1 CLUSTER DESCRIPTOR HEADER

The Header starts with the **wLength** field that contains the total number of bytes in the Cluster Descriptor, including the Cluster Descriptor Header and all included Cluster Descriptor Blocks.

The **wNrChannels** field indicates the number of audio channels present in the Cluster.

If for any reason, the Cluster is currently Inactive (not carrying any audio data), then its Cluster Descriptor shall not be used for any purpose.

Note: The Cluster Descriptor shall not generate an interrupt when the Cluster changes between Active and Inactive since the Cluster Descriptor itself does not change.

Table 4-3: Cluster Descriptor Header

Offset	Field	Size	Value	Description
0	wLength	2	Number	Size of the entire Cluster Descriptor, including the Header and all of its building blocks, in bytes.
2	wDescriptorType	2	Constant	EXT_INTERFACE Descriptor type.
4	wDescriptorSubtype	2	Constant	CLUSTER Descriptor subtype.
6	wDescriptorID	2	Number	Unique ID of this Cluster Descriptor.
8	wStrDescriptorID	2	Number	Unique ID of a String Descriptor.

Offset	Field	Size	Value	Description
10	wNrChannels	2	Number	Number of channels present in the Cluster: n.

4.4.2 CLUSTER DESCRIPTOR BLOCK

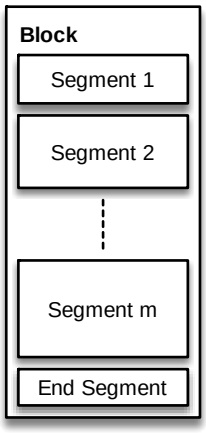
The Cluster Descriptor Header is followed by one or more Cluster Descriptor Blocks. There is an optional Common Block, followed by as many mandatory Channel Blocks as there are channels in the Cluster. Each Block consists of one or more Segments.

The Common Block consists of Segments that contain relevant information about characteristics of the Cluster as a whole. All Common Block Segments are optional.

A Channel Block consists of Segments that contain relevant information about that channel's characteristics. At a minimum, each Channel Block shall contain either an Information Segment or an Ambisonic Segment and a single End Block Segment. All other Segments are optional.

Figure 4-5 further illustrates the above concepts.

Figure 4-5: Cluster Descriptor Block



4.4.2.1 SEGMENTS

There are two types of Segments. Common Block Segments contain pertinent information about the Cluster as a whole. Channel Block Segments contain pertinent information about certain aspects of a particular channel. Both Segment types share the same layout.

The **wSegmentType** field describes the Segment Type (Common Block or Channel Block) and the type of content contained in the Segment.

The layout for a Segment is always as follows:

Table 4-4: Cluster Descriptor Segment

Offset	Field	Size	Value	Description
0	wLength	2	Number	Length of the Segment, in bytes: 4+n.

Offset	Field	Size	Value	Description
2	wSegmentType	2	Constant	Describes the Segment Type and the type of content in the Segment.
4	Segment-specific	n		Segment-specific content.

4.4.2.1.1 END BLOCK SEGMENT

Each Block is terminated by an End Block Segment. The End Block Segment marks the end of the variable length Block. The End Block Segment does not have a Segment-specific section and is structured as follows:

Table 4-5: End Block Segment

Offset	Field	Size	Value	Description
0	wLength	2	Number	Length of the Segment, in bytes: 4.
2	wSegmentType	2	Constant	CLUSTER_END_BLOCK.

4.4.2.1.2 COMMON BLOCK SEGMENTS

This specification currently does not define any Common Block Segments.

4.4.2.1.3 CHANNEL BLOCK SEGMENTS

The following Channel Block Segment types are defined:

- Information
- Ambisonic
- Channel Description

Values for the Channel Block Segment types can be found in Appendix A.13, “Cluster Descriptor Segment Types.”

4.4.2.1.3.1 INFORMATION SEGMENT

The Information Segment contains relevant information for a particular channel in the Cluster. It is mutually exclusive with the Ambisonic Segment for that same channel.

The **wChPurpose** field indicates the *primary* Purpose of the channel. Currently defined Purposes for a channel are:

- Generic Audio: contains audio primarily used for direct capture or reproduction.
- Voice: intended to be interpreted by humans.
- Speech: intended to be interpreted or generated by machine.
- Ambient: contains audio other than the primary channels.
- Reference: contains final processed audio. For example, a reference for AEC processing.
- Ultrasonic: contains signals with spectral content above audible limits (typically >20 kHz).
- Vibrokinetic: contains very low frequency information, typically used to actuate vibrators or motion actuators.
- Sense: contains current/voltage/temperature/etc. real-time data.
- Non-Audio: indicates that the channel carries non-audio information. Examples of non-audio information are real-time pressure sensing data or amplifier gain feedback data, etc.
- Silence: indicates that the channel contains inaudible content, i.e., silence.

The values for the Purposes listed above can be found in Appendix A.14, “Channel Purpose Definitions.”

Note that great care should be taken when indicating the primary Purpose for an audio channel. For example, if unprocessed audio is subsequently processed specifically to be used by applications that use human voice as input, such a channel may be marked Voice. Also, a channel that is marked for a particular primary Purpose may be used for other purposes, compatible with the primary Purpose, as well.

The **wChRelationship** field describes the relationship of this channel with respect to the other channels in the Cluster. Currently defined relationships for a channel are described in the table below.

Array
Pattern_X
Pattern _Y
Pattern _A
Pattern _B
Pattern _M
Pattern _S

Table 4-6: Channel Relationships

	Channel Relationship	Acronym	Description
Generic	RELATIONSHIP_UNDEFINED	UND	Relationship undefined or unknown
	Mono	M	A mono channel
	Left	L	A generic Left channel. Not intended for headphones
	Right	R	A generic Right channel. Not intended for headphones
	Array	AR	A channel that is part of an array configuration

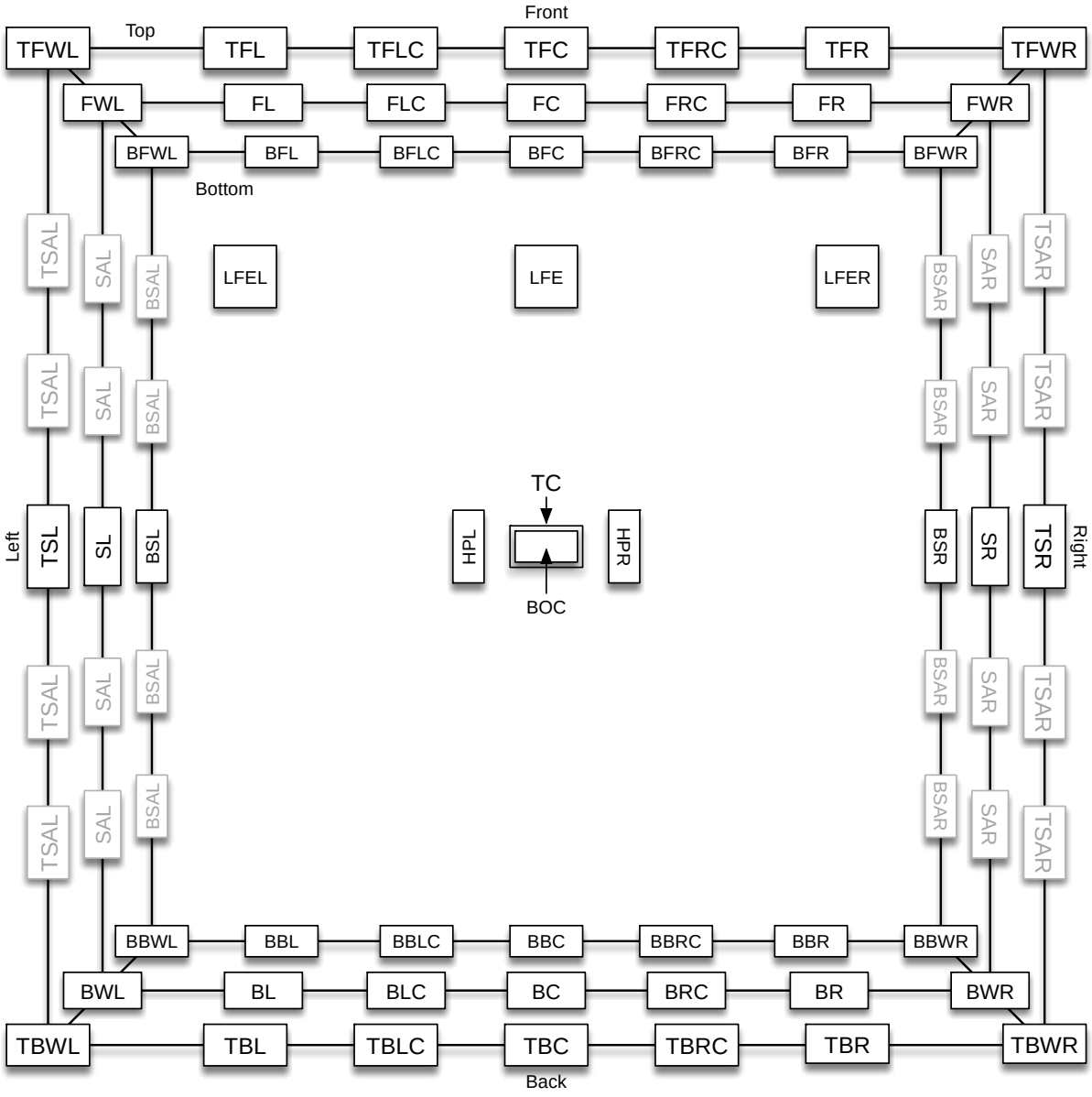
	Channel Relationship	Acronym	Description
Input-Related	Headset_Mic	HM	A Headset microphone
	Headset_Mic Left	HML	A Headset microphone to the left of the mouth
	Headset_Mic Right	HMR	A Headset microphone to the right of the mouth
	Headset_Mic Center	HMC	A Headset microphone in a central position w.r.t. the mouth
	Body_Mic	BM	A microphone worn on the body
	Body_Mic Left	BML	A microphone worn on the left side of the body
	Body_Mic Right	BMR	A microphone worn on the right side of the body
	Body_Mic Center	BMC	A microphone worn centrally on the body
	Lapel_Mic	LM	A microphone worn on the lapel
	Lapel_Mic Left	LML	A microphone worn on the left lapel
	Lapel_Mic Right	LMR	A microphone worn on the right lapel
	Lapel_Mic Center	LMC	A microphone worn centrally at lapel height
	Pattern_X	PX	The X channel of an X/Y microphone
	Pattern _Y	PY	The Y channel of an X/Y microphone
	Pattern _A	PA	The A channel of an A/B microphone
	Pattern _B	PB	The B channel of an A/B microphone
	Pattern _M	PM	The M channel of an M/S microphone
	Pattern _S	PS	The S channel of an M/S microphone
Input/Output-Related	Front Left	FL	Refer to Figure 4-6. Not intended for headphones
	Front Right	FR	Refer to Figure 4-6. Not intended for headphones
	Front Center	FC	Refer to Figure 4-6
	Front Left of Center	FLC	Refer to Figure 4-6
	Front Right of Center	FRC	Refer to Figure 4-6
	Front Wide Left	FWL	Refer to Figure 4-6
	Front Wide Right	FWR	Refer to Figure 4-6
	Side Left	SL	Refer to Figure 4-6
	Side Right	SR	Refer to Figure 4-6
	Surround Array Left	SAL	Refer to Figure 4-6
	Surround Array Right	SAR	Refer to Figure 4-6
	Back Left	BL	Refer to Figure 4-6
	Back Right	BR	Refer to Figure 4-6
	Back Center	BC	Refer to Figure 4-6
	Back Left of Center	BLC	Refer to Figure 4-6
	Back Right of Center	BRC	Refer to Figure 4-6
	Back Wide Left	BWL	Refer to Figure 4-6

	Channel Relationship	Acronym	Description
	Back Wide Right	BWR	Refer to Figure 4-6
	Top Center	TC	Refer to Figure 4-6
	Top Front Left	TFL	Refer to Figure 4-6
	Top Front Right	TFR	Refer to Figure 4-6
	Top Front Center	TFC	Refer to Figure 4-6
	Top Front Left of Center	TFLC	Refer to Figure 4-6
	Top Front Right of Center	TFRC	Refer to Figure 4-6
	Top Front Wide Left	TFWL	Refer to Figure 4-6
	Top Front Wide Right	TFWR	Refer to Figure 4-6
	Top Side Left	TSL	Refer to Figure 4-6
	Top Side Right	TSR	Refer to Figure 4-6
	Top Surround Array Left	TSAL	Refer to Figure 4-6
	Top Surround Array Right	TSAR	Refer to Figure 4-6
	Top Back Left	TBL	Refer to Figure 4-6
	Top Back Right	TBR	Refer to Figure 4-6
	Top Back Center	TBC	Refer to Figure 4-6
	Top Back Left of Center	TBLC	Refer to Figure 4-6
	Top Back Right of Center	TBRC	Refer to Figure 4-6
	Top Back Wide Left	TBWL	Refer to Figure 4-6
	Top Back Wide Right	TBWR	Refer to Figure 4-6
	Bottom Center	BOC	Refer to Figure 4-6
	Bottom Front Left	BFL	Refer to Figure 4-6
	Bottom Front Right	BFR	Refer to Figure 4-6
	Bottom Front Center	BFC	Refer to Figure 4-6
	Bottom Front Left of Center	BFLC	Refer to Figure 4-6
	Bottom Front Right of Center	BFRC	Refer to Figure 4-6
	Bottom Front Wide Left	BFWL	Refer to Figure 4-6
	Bottom Front Wide Right	BFWR	Refer to Figure 4-6
	Bottom Side Left	BSL	Refer to Figure 4-6
	Bottom Side Right	BSR	Refer to Figure 4-6
	Bottom Surround Array Left	BSAL	Refer to Figure 4-6
	Bottom Surround Array Right	BSAR	Refer to Figure 4-6
	Bottom Back Left	BBL	Refer to Figure 4-6
	Bottom Back Right	BBR	Refer to Figure 4-6
	Bottom Back Center	BBC	Refer to Figure 4-6

	Channel Relationship	Acronym	Description
	Bottom Back Left of Center	BBLC	Refer to Figure 4-6
	Bottom Back Right of Center	BBRC	Refer to Figure 4-6
	Bottom Back Wide Left	BBWL	Refer to Figure 4-6
	Bottom Back Wide Right	BBWR	Refer to Figure 4-6
	Low Frequency Effects	LFE	Refer to Figure 4-6
	Low Frequency Effects Left	LFEL	Refer to Figure 4-6
	Low Frequency Effects Right	LFER	Refer to Figure 4-6
	Headphone Left	HPL	Refer to Figure 4-6
	Headphone Right	HPR	Refer to Figure 4-6

The following figure presents a spatial “view from above” of the different Channel Relationships.

Figure 4-6: 3D Representation of the Channel Relationships



Constant names, acronyms, and their associated values are listed in Appendix A.15, “Channel Relationship Definitions.”

The **wChannelID** field is used to uniquely identify the channel within the scope of the entire Audio Function. No two channels in the entire Audio Function shall use the same value for this field. Setting the **wChannelID** field to zero is prohibited. Implementations shall populate this field so that a Host can retrieve meaningful signal routing information from within the Audio Function. Although it is in some cases difficult to establish definite rules on how to propagate Channel IDs when Clusters pass through certain types of Units, it is left to the implementation to decide how to manage Channel IDs in those cases.

The **wChGroupID** field is used to create a link among a subset of channels in the Cluster by specifying the same value in the **wChGroupID** field for those channels. This field is useful when multiple sets of related channels are

present in the same Cluster. For example, a single Cluster may carry one ChannelGroup of generic Left and Right stereo channels and another ChannelGroup of generic Left and Right stereo channels. The first generic Left and Right channels would share the same ChannelGroup ID value, indicating that they belong to a ChannelGroup, and the second generic Left and Right stereo channels would share another ChannelGroup ID value, indicating they belong to a different ChannelGroup. The scope of the **wChGroupID** field is limited to the Cluster to which the channels belong. A value of zero in this field indicates that the channel is not part of a ChannelGroup. Note also that this specification does not prohibit ChannelGroups that contain only a single channel. However, the ChannelGroup ID should not be confused with the concept of the Channel ID.

Each channel in a ChannelGroup shall have a

The **wConID** field identifies the Connector (through its Connector Entity) to which the channel is associated. If the channel is not associated with a Connector, then the **wConID** field shall be set to zero. See Section 4.5.3.14, “Connector Entity Descriptor” for more details.

Table 4-7: Information Segment

Offset	Field	Size	Value	Description
0	wLength	2	Number	Length of the Channel Information Segment, in bytes: 14.
2	wSegmentType	2	Constant	CHANNEL_INFORMATION.
4	wChPurpose	2	Constant	Intended purpose of the channel.
6	wChRelationship	2	Constant	Describes the relationship of this channel with the other channels. Refer to Appendix A.15, “Channel Relationship Definitions.”
8	wChannelID	2	Number	ID used to uniquely identify a channel.
10	wChGroupID	2	Number	ID used to group channels together.
12	wConID	2	Number	The Connector ID of the Connector to which the channel is associated.

4.4.2.1.3.2 AMBISONIC SEGMENT

The Ambisonic Segment contains relevant information for a particular Ambisonic channel in the Cluster. It is mutually exclusive with the Information Segment for that same channel.

The **wCompOrdering** field indicates the convention used for ordering of the spherical harmonics. See Appendix A.16, “Ambisonic Component Ordering Convention Types” for the supported component ordering conventions. All channels in a ChannelGroup (see below) shall indicate the same component ordering convention.

The **wACN** field contains the Ambisonic Channel Number.

The **wAmbNorm** field indicates the type of normalization used for the channel. See Appendix A.17, “Ambisonic Normalization Types” for the supported normalization types. All channels in a ChannelGroup (see below) shall indicate the same normalization type.

The **wChannelID** field is used to uniquely identify the channel within the scope of the entire Audio Function. No two channels in the entire Audio Function shall use the same non-zero value for this field. Setting the **wChannelID** field to zero is prohibited. Implementations shall populate this field so that a Host can retrieve meaningful signal routing information from within the Audio Function. Although it is in some cases difficult to establish definite rules

on how to propagate Channel IDs when Clusters pass through certain types of Units, it is left to the implementation to decide how to manage Channel IDs in those cases.

The **wChGroupID** field is used to create a link among a subset of channels in the Cluster by specifying the same value in the **wChGroupID** field for those channels. This field is useful when multiple sets of related channels are present in the same Cluster. For example, a single Cluster may carry a ChannelGroup of 5.1 channels and another ChannelGroup of microphone channels. The 5.1 channels would share the same ChannelGroup ID value, indicating that they belong to a (5.1) ChannelGroup, and the microphone channels would share another ChannelGroup ID value, indicating they belong to a different (microphone) ChannelGroup. The scope of the **wChGroupID** field is limited to the Cluster to which the channels belong. A value of zero in this field indicates that the channel is not part of a ChannelGroup. Note also that this specification does not prohibit ChannelGroup that contain only a single channel. However, the ChannelGroup ID should not be confused with the concept of the Channel ID.

The **wConID** field identifies the Connector (through its Connector Entity) to which the channel is associated. If the channel is not associated with a Connector, then the **wConID** field shall be set to zero. See Section 4.5.3.14, “Connector Entity Descriptor” for more details.

Table 4-8: Ambisonic Segment

Offset	Field	Size	Value	Description
0	wLength	2	Number	Length of the Segment, in bytes: 16.
2	wSegmentType	2	Constant	CHANNEL_AMBISONIC.
4	wCompOrdering	2	Number	The Component Ordering convention for the Ambisonic Spherical Harmonics. Refer to Appendix A.16, “Ambisonic Component Ordering Convention Types.”
6	wACN	2	Number	Ambisonic Channel Number.
8	wAmbNorm	2	Number	Applied channel normalization. Refer to Appendix A.17, “Ambisonic Normalization Types.”
10	wChannelID	2	Number	ID used to uniquely identify a channel.
12	wChGroupID	2	Number	ID used to group channels together.
14	wConID	2	Number	The Connector ID of the Connector to which the channel is associated.

4.4.2.1.3.3 CHANNEL DESCRIPTION SEGMENT

The Channel Description Segment contains the ID of a String Descriptor in the **wChStrDescrID** field that provides more information about the channel.

Table 4-9: Channel Description Segment

Offset	Field	Size	Value	Description
0	wLength	2	Number	Length of the Segment, in bytes: 6.
2	wSegmentType	2	Constant	CHANNEL_DESCRIPTION.
4	wChStrDescriptorID	2	Number	ID of a String Descriptor that further describes the channel.

4.4.3 EXAMPLE CLUSTER DESCRIPTOR

A Cluster that carries 5.1 discrete channels may have the following Cluster Descriptor:

Table 4-10: Cluster Descriptor Example

	Offset	Field	Size	Value	Description
Header	0	wLength	2	0x3E	Total length of the Descriptor, in bytes: 120.
	2	wDescriptorType	2	0x21	EXT_INTERFACE Descriptor type.
	4	wDescriptorSubtype	2	0x11	CLUSTER Descriptor subtype.
	6	wDescriptorID	2	Implementation-dependent	Unique ID of this Cluster Descriptor.
	8	wStrDescriptorID	2	Implementation-dependent	Unique ID of a String Descriptor.
	10	wNrChannels	2	6	Number of channels present in the Cluster.
Information Segment	12	wLength	2	0x08	Length of the Information Segment, in bytes.
	14	wSegmentType	2	0x20	CHANNEL_INFORMATION.
	16	wChPurpose	2	0x00	Generic Audio.
	18	wChRelationship	2	0x0B	Front Left.
	20	wChannelID	2	0x0020	Channel ID.
	22	wChGroupID	2	0x0001	ID used to group channels together.
	24	wConID	2	0x0000	No associated Connector Entity.
End	26	wLength	2	0x03	Length of the End Block Segment, in bytes.
	28	wSegmentType	2	0xFF	END_SEGMENT.
Information	30	wLength	2	0x08	Length of the Information Segment, in bytes.
	32	wSegmentType	2	0x20	CHANNEL_INFORMATION.
	34	wChPurpose	2	0x00	Generic Audio.
	36	wChRelationship	2	0x0C	Front Right.
	38	wChannelID	2	0x0021	Channel ID.
	40	wChGroupID	2	0x0001	ID used to group channels together.
	42	wConID	2	0x0000	No associated Connector Entity.
End	44	wLength	2	0x03	Length of the End Block Segment, in bytes.
	46	wSegmentType	2	0xFF	END_SEGMENT.
Information	48	wLength	2	0x08	Length of the Information Segment.
	50	wSegmentType	2	0x20	CHANNEL_INFORMATION.
	52	wChPurpose	2	0x00	Generic Audio.
	54	wChRelationship	2	0x0D	Front Center.
	56	wChannelID	2	0x0022	Channel ID.
	58	wChGroupID	2	0x0001	ID used to group channels together.
	60	wConID	2	0x0000	No associated Connector Entity.

	Offset	Field	Size	Value	Description
End	62	wLength	2	0x03	Length of the End Block Segment, in bytes.
	64	wSegmentType	2	0xFF	END_SEGMENT.
Information	66	wLength	2	0x08	Length of the Information Segment.
	68	wSegmentType	2	0x20	CHANNEL_INFORMATION.
	70	wChPurpose	2	0x00	Generic Audio.
	72	wChRelationship	2	0x14	Surround Array Left.
	74	wChannelID	2	0x0023	Channel ID.
	76	wChGroupID	2	0x0001	ID used to group channels together.
	78	wConID	2	0x0000	No associated Connector Entity.
End	80	wLength	2	0x03	Length of the End Block Segment.
	82	wSegmentType	2	0xFF	END_SEGMENT.
Information	84	wLength	2	0x08	Length of the Information Segment.
	86	wSegmentType	2	0x20	CHANNEL_INFORMATION.
	88	wChPurpose	2	0x00	Generic Audio.
	90	wChRelationship	2	0x15	Surround Array Right.
	92	wChannelID	2	0x0024	Channel ID.
	94	wChGroupID	2	0x0001	ID used to group channels together.
	96	wConID	2	0x0000	No associated Connector Entity.
End	98	wLength	2	0x03	Length of the End Block Segment.
	100	wSegmentType	2	0xFF	END_SEGMENT.
Information	102	wLength	2	0x08	Length of the Information Segment.
	104	wSegmentType	2	0x20	CHANNEL_INFORMATION.
	106	wChPurpose	2	0x00	Generic Audio.
	108	wChRelationship	2	0x43	Low Frequency Effects.
	110	wChannelID	2	0x0025	Channel ID.
	112	wChGroupID	2	0x0001	ID used to group channels together.
	114	wConID	2	0x0000	No associated Connector Entity.
End	116	wLength	2	0x03	Length of the End Block Segment.
	118	wSegmentType	2	0xFF	END_SEGMENT.

4.4.4 CEA-861.2 CHANNEL MAPPING

The Advanced Audio Extensions specification CEA-861.2 allocates speaker spatial locations in a different order than what is defined in this USB Audio Definition. Also, there are channel relationships defined that do not appear in the USB Audio Definition. To provide consistency in implementations that use both the CEA-861.2 allocations and the USB Audio channel relationships, a mapping scheme between the two is included here to which such implementations shall adhere.

The mapping between the USB-defined channel relationships and the CEA speaker allocations is included in the table in Appendix A.15, “Channel Relationship Definitions.”

4.4.5 PHYSICAL VERSUS LOGICAL CLUSTER

This specification makes a distinction between a logical and physical Cluster. Hence, there are also two types of Cluster Descriptors defined:

- Logical Cluster Descriptor
- Physical Cluster Descriptor

The layout of these Descriptors is identical as described in Section above.

Both the logical and physical Cluster Descriptors are not independent Descriptors as such. They are always referenced by other Descriptors. The referencing Descriptors always include a **wClusterDescrID** field that contains the unique ID of the Cluster Descriptor they reference.

The following Descriptors reference a logical Cluster Descriptor:

- Input Terminal Descriptor
- Mixer Unit Descriptor
- Processing Unit Descriptor
- Extension Unit Descriptor

The class-specific AudioStreaming Interface Descriptor in each Alternate Setting of an AudioStreaming Interface (except for Alternate Setting 0) reference a physical Cluster Descriptor.

Connectors Descriptors also reference a physical Cluster Descriptor to provide information about the physical channels that travel over the various connectors, associated with a Terminal.

4.5 AUDIOCONTROL INTERFACE DESCRIPTORS

The AudioControl (AC) Interface Descriptors contain all relevant information to fully characterize the corresponding Audio Function. The standard Interface Descriptor characterizes the Interface itself, whereas the class-specific Interface Descriptors provide pertinent information concerning the internals of the Audio Function. It specifies revision level information and lists the capabilities of each Entity within the Audio Function.

4.5.1 STANDARD AC INTERFACE DESCRIPTOR

The standard AC Interface Descriptor is identical to the standard interface Descriptor defined in the *USB Core Specifications*, except that some fields now have dedicated values.

Table 4-11: Standard AC Interface Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	Number	Size of this Descriptor, in bytes: 9.
1	bDescriptorType	1	Constant	INTERFACE Descriptor type.
2	bInterfaceNumber	1	Number	Number of Interface. A zero-based value identifying the index in the array of concurrent interfaces supported by this configuration.

Offset	Field	Size	Value	Description
3	bAlternateSetting	1	Number	Value used to select an Alternate Setting for the Interface identified in the prior field. Shall be set to 0.
4	bNumEndpoints	1	Number	Number of Endpoints used by this interface (excluding Endpoint 0). This number is either 0 or 1 if the optional interrupt Endpoint is present.
5	bInterfaceClass	1	AUDIO	Audio Interface Class code (assigned by the USB). See Appendix A.5, “Audio Interface Class Code.”
6	bInterfaceSubClass	1	AUDIO CONTROL	Audio Interface Subclass code. Assigned by this specification. See Appendix A.6, “Audio Interface Subclass Codes.”
7	bInterfaceProtocol	1	IP_VERSION_04_00	Interface Protocol code. Indicates the current version of the specification. See Appendix A.7, “Audio Interface Protocol Codes”
8	iInterface	1	Index	Index of a String Descriptor that describes this interface. Zero indicates that no string Descriptor is associated with this Descriptor.

4.5.2 CLASS-SPECIFIC AC INTERFACE DESCRIPTOR

The class-specific AC Interface Descriptor is a traditional Descriptor that contains the list of Extended Descriptor IDs, used to fully describe the AudioControl Interface and the Audio building blocks it encapsulates. All class-specific Clock Descriptor IDs, all Unit Descriptors ID, all Terminal Descriptor IDs, and all Power Domain Entity Descriptor IDs shall be in that list. Cluster Descriptor IDs, Terminal Companion Descriptor IDs, and Connector Descriptor IDs shall not be included in the list as they are indirectly referenced by other Extended Descriptors in the list.

The order in which the Entity Descriptor IDs are reported is not important because every Descriptor can be identified through its **bDescriptorType** and **bDescriptorSubtype** field.

The following table defines the class-specific AC Interface Descriptor.

Table 4-12: Class-specific AC Interface Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	Number	Size of this Descriptor, in bytes: 4+n*2.
1	bDescriptorType	1	Constant	CS_INTERFACE Descriptor type.
2	bDescriptorSubtype	1	Constant	AC_GENERIC Descriptor subtype.
3	bNrDescriptorIDs	1	Number	Number of Descriptor IDs in the list: n.
4	waDescriptorID(1)	2	Bitmap	First Descriptor ID.
…	…	…	…	…
4+(n-1)*2	waDescriptorID(n)	2	Bitmap	Last Descriptor ID.

If there is a need to list more than 125 Descriptor IDs, then more Descriptors of subtype AC_GENERIC may be included.

4.5.3 CLASS-SPECIFIC AC INTERFACE BUILDING BLOCK DESCRIPTORS

There are three types of Clock Entities, seven types of Units, two types of Terminals, and one type of Power Domain Entity. Their respective Extended Descriptors are described in the following sections.

4.5.3.1 AC SELF DESCRIPTOR

The Self Descriptor provides information to the Host that is related to the functional aspects of the AudioControl Interface itself. It indicates the presence of optional AudioControls at the Interface level.

The presence of the optional Latency Controls is advertised here in the **dOptControls** field of the class-specific AudioControl Interface Descriptor and not repeated in every Terminal and Unit Descriptor. If implemented, every Terminal and Unit within the Device shall expose a Latency Control.

Bit D1 in the **dOptControls** field indicates whether the Commit Capability supports Commits by CommitGroup (D1 = 0b1) or only Function-wide Commits (D1 = 0b0).

The following table defines the Self Descriptor.

Table 4-13: Self Descriptor

Offset	Field	Size	Value	Description
0	wLength	2	Number	Size of this Descriptor, in bytes: 14.
2	wDescriptorType	2	Constant	EXT_INTERFACE Descriptor type.
4	wDescriptorSubtype	2	Constant	AC_SELF Descriptor subtype.
6	wDescriptorID	2	Number	Unique ID for this Descriptor.
8	wStrDescriptorID	2	Number	Unique ID for a String Descriptor.
10	dOptControls	4	Bitmap	D0: Latency Controls present. D1: Commit Capability supports update by CommitGroup. D31..1: Reserved.

4.5.3.2 INPUT TERMINAL DESCRIPTOR

The Input Terminal Descriptor provides information to the Host that is related to the functional aspects of the Input Terminal.

The Input Terminal is uniquely identified by the value in the **wTerminalID** field. This value shall be passed in the **wEntityID** field of each Command that is directed to the Terminal.

The **wCSourceID** field contains a constant indicating to which Clock Entity the Clock Input Pin of this Input Terminal is connected. If the Clock Input Pin is not connected to a Clock Entity, the **wCSourceID** field shall be set to zero.

The **wPCC** field contains the Pin Channel Count value for the connection that originates at the Output Pin of the Input Terminal.

The **waClusterDescrID()** array contains the IDs of the Cluster Descriptors that characterize the Clusters that may be exposed on the Output Pin of the Input Terminal. The **wNrClusterDescrIDs** field contains the number of elements

in that array and shall always be greater than zero. For a detailed description of the Cluster Descriptor, see Section 4.4, “Cluster Descriptor”.

In case the Input Terminal represents an AudioStreaming Interface, then the Cluster Control shall be present (**dOptControls** bit D0 = 0b1) and be implemented as Read-Only. The current Cluster Configuration shall be determined by the currently selected Alternate Setting of the AudioStreaming Interface, and the Cluster Control CUR value shall be set to the ordinal number of the currently selected Alternate Setting of the AudioStreaming Interface (including zero when Alternate Setting zero is selected). The **wNrClusterDescrIDs** field shall be set to the number of Alternate Settings the AudioStreaming Interface supports (excluding Alternate Setting zero) and the **waClusterDescrID()** array shall list in order, the IDs of the Cluster Descriptors that correspond to each of the active Alternate Settings of the AudioStreaming Interface.

In case the Input Terminal represents either an external connection via one or more Connector Entities or an internal transducer, then, if more than one Cluster Configuration is available, the Cluster Control shall be present (**dOptControls** bit D0 = 0b1) and be implemented as Read-Write. If only one Cluster Configuration is available, the Cluster Control shall not be present (**dOptControls** bit D0 = 0b0).

The Cluster Active Control shall always be present (and is therefore not part of the **dOptControls** field).

The **wTermCompDescrID** field contains the unique ID of the Terminal Companion Descriptor that is associated with this Terminal. This field shall be set to zero (no Terminal Companion Descriptor available) when the Input Terminal represents an AudioStreaming OUT Interface. For a detailed description of the Terminal Companion Descriptor, see Section 4.5.3.4, “Terminal Companion Descriptor.”

If an AudioStreaming Interface is associated with this Terminal, then Variant 1 of the Descriptor applies and the **wDescriptorVariant** field shall be set to VARIANT_INTERFACE. The **bInterfaceNumber** field shall contain the interface number of that AudioStreaming Interface.

If one or more Connector Entities are associated with the Terminal, then Variant 2 of the Descriptor applies and the **wDescriptorVariant** field shall be set to VARIANT_ENTITIES. The **wNrAssocEntityIDs** field shall contain the number of elements in the following **waAssocEntityID()** array that contains the unique IDs of those Connector Entities.

If there are no Entities associated with this Terminal, then the **wDescriptorVariant** field shall be set to VARIANT_NONE and no Descriptor Variant shall be present.

The OCN:ICN:IPN triplet used to access an AudioControl within the Input Terminal shall be set to 0:0:0.

The following table presents an outline of the Input Terminal Descriptor.

Table 4-14: Input Terminal Descriptor

Offset	Field	Size	Value	Description
0	wLength	2	Number	Size of this Descriptor, in bytes: • VARIANT_NONE: 26+p*2. • VARIANT_INTERFACE: 27+p*2. • VARIANT_ENTITIES: 28+p*2+q*2.
2	wDescriptorType	2	Constant	EXT_INTERFACE Descriptor type.
4	wDescriptorSubtype	2	Constant	INPUT_TERMINAL Descriptor subtype.
6	wDescriptorID	2	Number	Unique ID for this Descriptor.

Offset	Field	Size	Value	Description
8	wStrDescriptorID	2	Number	Unique ID for a String Descriptor.
10	wTerminalID	2	Number	Value uniquely identifying the Terminal within the Audio Function. This value is used in all Commands to address this Terminal.
12	wCSourceID	2	Number	ID of the Clock Entity to which this Input Terminal is connected.
14	dOptControls	4	Bitmap	D0: Cluster Control. D1: Voltage Control D2: Overload Control. D3: Underflow Control. D4: Overflow Control. D31..5: Reserved.
18	wPCC	2	Number	The Pin Channel Count for the connection originating at the Output Pin of this Input Terminal.
20	wNrClusterDescrIDs	2	Number	Number of elements in the **waClusterDescrID()** array: p.
22	waClusterDescrID(1)	2	Number	ID of the first Cluster Descriptor for this Input Terminal.
…	…	…	…	…
20+p*2	waClusterDescrID(p)	2	Number	ID of the last Cluster Descriptor for this Input Terminal.
22+p*2	wTermCompDescrID	2	Number	ID of the Terminal Companion Descriptor for this Input Terminal. Shall be set to zero if no Terminal Companion Descriptor is present.
24+p*2	wDescriptorVariant	2	Number	Variant number of the following Descriptor Variant. Shall be either VARIANT_NONE, VARIANT_INTERFACE or VARIANT_ENTITIES.

Variant VARIANT_INTERFACE applies when the Input Terminal is associated with an AudioStreaming Interface.

Offset	Field	Size	Value	Description
26+p*2	bInterfaceNumber	1	Number	Interface number of the associated AudioStreaming Interface.

Variant VARIANT_ENTITIES applies when the Input Terminal is associated with one or more Connector Entities.

Offset	Field	Size	Value	Description
26+p*2	wNrAssocEntityIDs	2	Number	Number of elements in the **baAssocEntityID()** array: q.
26+p*2+2	waAssocEntityID(1)	2	Number	ID of the first Entity associated with this Input Terminal.
…	…	…	…	…
26+p*2+q*2	waAssocEntityID(q)	2	Number	ID of the last Entity associated with this Input Terminal.

4.5.3.3 OUTPUT TERMINAL DESCRIPTOR

The Output Terminal Descriptor provides information to the Host that is related to the functional aspects of the Output Terminal.

The Output Terminal is uniquely identified by the value in the **wTerminalID** field. This value shall be passed in the **wEntityID** field of each Command that is directed to the Terminal.

The **wSourceID** field is used to describe the connectivity for this Terminal. It contains the ID of the Unit or Terminal to which this Output Terminal is connected via its Input Pin. The Cluster Descriptor, describing the logical channels entering the Output Terminal is not repeated here. It is up to the Host software to trace the connection 'upstream' to locate the Cluster Descriptor pertaining to this Cluster.

The **wCSourceID** field contains a constant indicating to which Clock Entity the Clock Input Pin of this Output Terminal is connected. If the Clock Input Pin is not connected to a Clock Entity, the **wCSourceID** field shall be set to zero**.**

The PCC value for the Input Pin of the Output Terminal is not included here and is inherited from the first upstream Entity that defines a PCC value on its Output Pin.

The **wTermCompDescrID** field contains the unique ID of the Terminal Companion Descriptor that is associated with this Terminal. This field shall be set to zero (no Terminal Companion Descriptor available) when the Output Terminal represents an AudioStreaming IN Interface. For a detailed description of the Terminal Companion Descriptor, see Section 4.5.3.4, "Terminal Companion Descriptor."

If an AudioStreaming Interface is associated with this Terminal, then the AudioStreaming Variant of the Descriptor applies and the **wDescriptorVariant** field shall be set to VARIANT_INTERFACE. The **bInterfaceNumber** field shall contain the interface number of that AudioStreaming Interface.

If one or more Connector Entities are associated with the Terminal, then the Entities Variant of the Descriptor applies and the **wDescriptorVariant** field shall be set to VARIANT_ENTITIES. The **wNrAssocEntityIDs** field shall contain the number of elements in the following **waAssocEntityID()** array that contains the unique IDs of those Connector Entities.

If there are no Entities associated with this Terminal, then the **wDescriptorVariant** field shall be set to VARIANT_NONE and no Descriptor Variant shall be present.

The OCN:ICN:IPN triplet used to access an AudioControl within the Output Terminal shall be set to 0:0:0.

The following table presents an outline of the Output Terminal Descriptor.

Table 4-15: Output Terminal Descriptor

Offset	Field	Size	Value	Description
0	wLength	2	Number	Size of this Descriptor, in bytes: • VARIANT_NONE: 24. • VARIANT_INTERFACE: 25. • VARIANT_ENTITIES: 24+q*2.
2	wDescriptorType	2	Constant	EXT_INTERFACE Descriptor type.
4	wDescriptorSubtype	2	Constant	OUTPUT_TERMINAL Descriptor subtype.
6	wDescriptorID	2	Number	Unique ID for this Descriptor.
8	wStrDescriptorID	2	Number	Unique ID for a String Descriptor.

Offset	Field	Size	Value	Description
10	wTerminalID	2	Number	Value uniquely identifying the Terminal within the Audio Function. This value is used in all Commands to address this Terminal.
12	wSourceID	2	Number	ID of the Unit or Terminal to which this Terminal is connected.
14	wCSourceID	2	Number	ID of the Clock Entity to which this Output Terminal is connected.
16	dOptControls	4	Bitmap	D0: Overload Control. D1: Underflow Control. D2: Overflow Control. D31..3: Reserved.
20	wTermCompDescrID	2	Number	ID of the Terminal Companion Descriptor for this Output Terminal. Shall be set to zero if no Terminal Companion Descriptor is present.
22	wDescriptorVariant	2	Number	Variant number of the following Descriptor Variant. Shall be either VARIANT_NONE, VARIANT_INTERFACE or VARIANT_ENTITIES.

Variant VARIANT_INTERFACE applies when the Output Terminal is associated with an AudioStreaming Interface.

Offset	Field	Size	Value	Description
24	bInterfaceNumber	1	Number	Interface number of the associated AudioStreaming Interface.

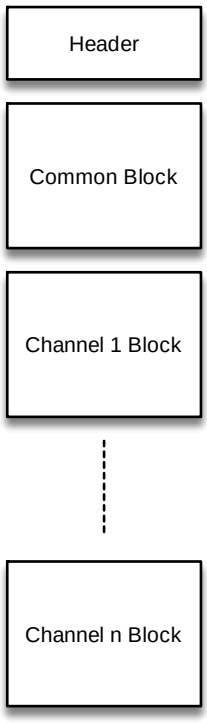
Variant VARIANT_ENTITIES applies when the Output Terminal is associated with one or more Connector Entities.

Offset	Field	Size	Value	Description
24	wNrAssocEntityIDs	2	Number	Number of elements in the **baAssocEntityID()** array: q.
26	waAssocEntityID(1)	2	Number	ID of the first Entity associated with this Output Terminal.
…	…	…	…	…
24+q*2	waAssocEntityID(q)	2	Number	ID of the last Entity associated with this Output Terminal.

4.5.3.4 TERMINAL COMPANION DESCRIPTOR

The Terminal Companion Descriptor optionally returns additional physical information about the channels that enter or leave the Terminal.

Figure 4-7: Terminal Companion Descriptor



The Terminal Companion Descriptor consists of a fixed Header, followed by an optional Common Block, followed by as many Channel Blocks as there are channels in the Cluster of the Terminal.

4.5.3.4.1 TERMINAL COMPANION DESCRIPTOR HEADER

The Header starts with the **wLength** field that contains the number of bytes in the Terminal Companion Descriptor Header.

The **wTotalLength** field contains the number of bytes in the entire Terminal Companion Descriptor, including the Header and all Terminal Companion Descriptor Blocks.

The **wNrChannels** field indicates the number of audio channels present in the logical Cluster of the Terminal.

Table 4-16: Terminal Companion Descriptor Header

Offset	Field	Size	Value	Description
0	wLength	2	Number	Length of the Terminal Companion Descriptor Header, in bytes: 14.
2	wDescriptorType	2	Constant	EXT_INTERFACE Descriptor type.
4	wDescriptorSubtype	2	Constant	TERMINAL_COMPANION Descriptor subtype.
6	wDescriptorID	2	Number	Unique ID of this Terminal Companion Descriptor.
8	wStrDescriptorID	2	Number	Unique ID of String Descriptor.

Offset	Field	Size	Value	Description
10	wTotalLength	2	Number	Total length of the Terminal Companion Descriptor, including the Terminal Companion Descriptor Header and all included Terminal Companion Descriptor Blocks.
12	wNrChannels	2	Number	Number of channels present in the Cluster of the Terminal.

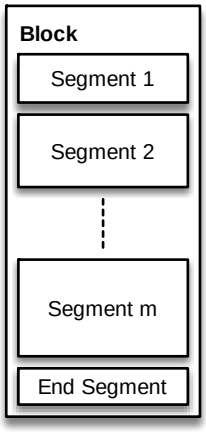
4.5.3.4.2 TERMINAL COMPANION DESCRIPTOR BLOCK

The Terminal Companion Descriptor Header is followed by one or more Terminal Companion Descriptor Blocks. There is an optional Common Block, followed by as many Channel Blocks as there are channels in the Cluster of the Terminal. Each Block consists of one or more Segments.

The Common Block consists of Segments that contain relevant information about characteristics of the Terminal as a whole. All Common Block Segments are optional.

A Channel Block consists of Segments that contain relevant information about that channel's physical characteristics. All Channel Block Segments are optional. It is strongly recommended that the same layout is used for each Channel Block, i.e., the same Segments appear in the same order in each Channel Block. Figure 4-8 further illustrates the above concepts.

Figure 4-8: Terminal Companion Channel Block



4.5.3.4.3 SEGMENTS

There are two types of Segments. Common Block Segments contain pertinent information about the Terminal as a whole. Channel Block Segments contain pertinent information about certain aspects of a particular channel in the Cluster of the Terminal. Both Segment types share the same layout.

The **wSegmentType** field describes the Segment Type (Common Block or Channel Block) and the type of content contained in the Segment.

The layout for a Segment is always as follows:

Table 4-17: Terminal Companion Descriptor Segment

Offset	Field	Size	Value	Description
0	wLength	2	Number	Length of the Segment, in bytes: 4+n.
2	wSegmentType	2	Constant	Describes the Segment Type and the type of content in the Segment.
4	Segment-specific	n		Segment-specific content.

4.5.3.4.3.1 END BLOCK SEGMENT

Each Block is terminated by an End Block Segment. The End Block Segment marks the end of the variable length Block. The End Block Segment does not have a Segment-specific section and is structured as follows:

Table 4-18: End Block Segment

Offset	Field	Size	Value	Description
0	wLength	2	Number	Length of the Segment, in bytes: 4.
2	wSegmentType	2	Constant	TERMINAL_COMPANION_END_BLOCK.

4.5.3.4.3.2 COMMON BLOCK SEGMENTS

The following Common Block Segment types are defined:

- EN 50332-2 Acoustic level
- EN 50332-2 Voltage level

Values for the Common Block Segment types can be found in Appendix A.18, “Terminal Companion Segment Types.”

4.5.3.4.3.2.1 EN 50332-2 ACOUSTIC LEVEL SEGMENT

The EN 50332-2 Acoustic Level Segment contains, in the **wOutputlevel** field, the measured headphone transducer output level in units of dB SPL A-weighted when all controls affecting the output are set to maximize the output level, following the procedures set forth within EN 50332-2:2013. The value may range from 0.00 dB(A) to +255.9961 dB(A) (0xFFFF) in steps of 1/256 dB(A) or 0.00390625 dB(A) (0x0001).

Only Output Terminals that represent *headphone transducers* shall be permitted to have this segment.

See [EN 50332-2:2013] for more details.

Table 4-19: EN 50332-2 Acoustic Level Segment

Offset	Field	Size	Value	Description
0	wLength	2	Number	Length of the Segment, in bytes: 6.
2	wSegmentType	2	Constant	EN_50332-2_SPL_LEVEL.
4	wOutputLevel	2	Number	Maximum achievable output level expressed in dB SPL (A-weighted) when rendering the EN 50332-2:2013 test signal.

4.5.3.4.3.2.2 EN 50332-2 VOLTAGE LEVEL SEGMENT

The EN 50332-2 Voltage Level Segment contains, in the **wOutputlevel** field, the measured headphone jack output level in units of millivolts rms (mV_{rms}) unweighted, when all controls affecting the output are set to maximize the output level, following the procedures set forth within EN 50332-2:2013. The value may range from 0.00 mV_{rms} to +65,635 mV_{rms} (0xFFFF) in steps of 1 mV_{rms} (0x0001).

Only Output Terminals that carry *electrical analog audio signals* shall be permitted to have this segment:

See [EN 50332-2:2013] for more details.

Table 4-20: EN 50332-2 Voltage Level Segment

Offset	Field	Size	Value	Description
0	wLength	2	Number	Length of the Segment, in bytes: 6.
2	wSegmentType	2	Constant	EN_50332-2_VOLTAGE_LEVEL.
4	wOutputLevel	2	Number	Maximum achievable output voltage, expressed in unweighted mV_{rms}, when rendering the EN 50332-2:2013 test signal.

4.5.3.4.3.3 CHANNEL BLOCK SEGMENTS

The following Channel Block Segment types are defined:

- Bandwidth
- Magnitude Response
- Magnitude/Phase Response
- Position

Values for the Channel Block Segment types can be found in Appendix A.18, “Terminal Companion Segment Types.”

4.5.3.4.3.3.1 BANDWIDTH SEGMENT

The Bandwidth Segment contains basic information about the audio bandwidth in the channel. The bandwidth is specified by providing the lower and upper -3 dB frequency points, in Hz, of the available band in the **dMinFreq** and **dMaxFreq** fields.

Table 4-21: Bandwidth Segment

Offset	Field	Size	Value	Description
0	wLength	2	Number	Length of the Segment, in bytes: 12.
2	wSegmentType	2	Constant	CHANNEL_BANDWIDTH.
4	dMinFreq	4	Number	Lower -3 dB frequency point.
8	dMaxFreq	4	Number	Upper -3 dB frequency point.

4.5.3.4.3.3.2 MAGNITUDE RESPONSE

The Magnitude Response Segment contains detailed information about the magnitude of the transfer function (frequency response) of the channel. The magnitude is specified as an array of [frequency point, magnitude] pairs. The frequency values are specified in Hz and the magnitude values may range from +127.9961 dB (0x7FFF) down

to -127.9961 dB (0x8001) in steps of 1/256 dB or 0.00390625 dB (0x0001). In addition, code 0x8000, representing silence (i.e., -∞ dB), may be used as well.

Table 4-22: Magnitude Segment

Offset	Field	Size	Value	Description
0	wLength	2	Number	Length of the Segment, in bytes: 10+n*6.
2	wSegmentType	2	Constant	CHANNEL_MAGNITUDE_RESPONSE.
4	wNrFreqPoints	2	Number	The number of frequency points, n.
6	dFreq(1)	4	Number	First frequency value.
10	wMagnitude(1)	2	Number	First magnitude value.
…	…	…	…	
4+n*6	dFreq(n)	4	Number	Last frequency value.
8+n*6	wMagnitude(n)	2	Number	Last magnitude value.

4.5.3.4.3.3.3 MAGNITUDE/PHASE RESPONSE

The Magnitude/Phase Response Segment contains detailed information about the magnitude and phase of the transfer function (frequency response) of the channel. The magnitude/phase is specified as an array of [frequency point, magnitude, phase] triplets. The frequency values are specified in Hz. The magnitude values may range from +127.9961 dB (0x7FFF) down to -127.9961 dB (0x8001) in steps of 1/256 dB or 0.00390625 dB (0x0001). In addition, code 0x8000, representing silence (i.e., -∞ dB), may be used as well. The phase values may range from +0.99996948242 * π down to -π (0x8000) in steps of 1/32768 * π (0x0001).

Table 4-23: Magnitude/Phase Segment

Offset	Field	Size	Value	Description
0	wLength	2	Number	Length of the Segment, in bytes: 10+n*8.
2	wSegmentType	2	Constant	CHANNEL_MAGNITUDE/PHASE_RESPONSE.
4	wNrFreqPoints	2	Number	The number of frequency points, n.
6	dFreq(1)	4	Number	First frequency value.
10	wMagnitude(1)	2	Number	First magnitude value.
12	wPhase(1)	2	Number	First phase value.
…	…	…	…	
4+n*8	dFreq(n)	4	Number	Last frequency value.
6+n*8	wMagnitude(n)	2	Number	Last magnitude value.
8+n*8	wPhase(n)	2	Number	Last phase value.

4.5.3.4.3.3.4 POSITION_XYZ SEGMENT

The Position_XYZ Segment contains the (X, Y, Z) Cartesian coordinates of the source or sink associated with the channel. The X, Y, and Z values are expressed in micrometers (µm) and are relative to an unspecified origin at (0, 0, 0). The Audio Function may have out-of-band means to indicate to the Host where the actual origin of the coordinate system is located on the Device. The value 0xFFFFFFFF has special meaning. It is used to indicate that the coordinate in a particular dimension is unknown.

Table 4-24: Position_XYX Segment

Offset	Field	Size	Value	Description
0	wLength	2	Number	Length of the Segment, in bytes: 16.
2	wSegmentType	2	Constant	CHANNEL_POSITION_XYZ.
4	dX	4	Number	X-coordinate of the audio source or sink, associated with the channel.
8	dY	4	Number	Y-coordinate of the audio source or sink, associated with the channel.
12	dZ	4	Number	Z-coordinate of the audio source or sink, associated with the channel.

4.5.3.4.3.3.5 POSITION_RΘΦ SEGMENT

The Position_ RΘΦ Segment contains the (R, Θ, Φ) spherical coordinates of the source or sink associated with the channel. The R, Θ, and Φ values are expressed in µm, µrad and µrad respectively, and are relative to an unspecified origin at (0, 0, 0). The Audio Function may have out-of-band means to indicate to the Host where the actual origin of the coordinate system is located on the Device. The value 0xFFFFFFFF has special meaning. It is used to indicate that the coordinate in a particular dimension is unknown.

Table 4-25: Position_RΘΦ Segment

Offset	Field	Size	Value	Description
0	wLength	2	Number	Length of the Segment, in bytes: 16.
2	wSegmentType	2	Constant	CHANNEL_POSITION_ R_THETA_PHI.
4	dR	4	Number	R-coordinate of the audio source or sink, associated with the channel.
8	dΘ	4	Number	Θ-coordinate of the audio source or sink, associated with the channel.
12	dΦ	4	Number	Φ-coordinate of the audio source or sink, associated with the channel.

4.5.3.5 MIXER UNIT DESCRIPTOR

The Mixer Unit is uniquely identified by the value in the **wUnitID** field of the Mixer Unit Descriptor. This value shall be passed in the **wEntityID** field of each Command that is directed to the Mixer Unit.

The **wNrInputPins** field contains the number of Input Pins (P) of the Mixer Unit. This equals the number of Clusters that enter the Mixer Unit. The connectivity of the Input Pins is described via the **waSourceID()** array, containing P elements. The index p into the array is one-based and directly related to the Input Pin numbers. **waSourceID(p)** contains the ID of the Unit or Terminal to which Input Pin p is connected.

The Cluster Descriptors, describing the logical channels entering the Mixer Unit are not repeated here. It is up to the Host software to trace the connections ‘upstream’ to locate the Cluster Descriptors pertaining to the Clusters and to determine the number of logical channels n_p in each incoming $Cluster_p$.

The PCC value for each of the Input Pins of the Mixer Unit is inherited from the first upstream Entity that defines a PCC value on its Output Pin.

The **wPCC** field contains the Pin Channel Count value for the connection that originates at the Output Pin of the Mixer Unit. This also determines the number of Output Pin channels the Mixer Unit supports.

The Input Pin Number (IPN), the Input Channel Number (ICN) and the Output Channel Number (OCN) are used to access a Mixer Control within the Mixer Unit and are defined as follows:

If P is the number of Input Pins, and N_p is the PCC for Input Pin p, and M is the PCC for the Output Pin, then the Input Pin Number IPN, the Input Channel Number ICN_p, and the Output Channel Number OCN for the Mixer Control residing on the crossing of Input Pin p, Input Pin channel q and Output Pin channel m is:

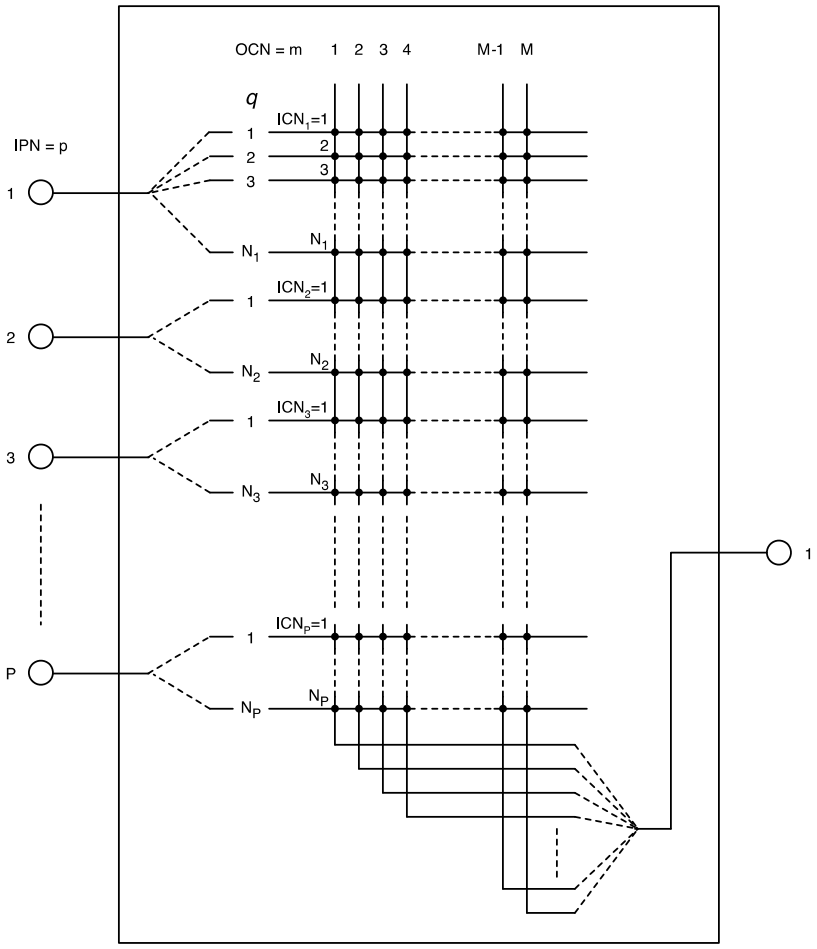
$$IPN = p; ICN_p = q; \text{and } OCN = m$$

where $(1 \leq p \leq P)$, $(1 \leq q \leq N_p)$, $(1 \leq m \leq M)$.

Note that if $n_p < N_p$, interaction with an AudioControl that resides on an Input Pin channel outside the range $[0..n_p]$ is still possible, i.e. the accessibility of an AudioControl shall not depend on the current Cluster Configuration.

The following figure presents a more graphical explanation.

Figure 4-9: Mixer internals



Because the Mixer Unit may redefine the spatial locations of the logical output channels, contained in its output Cluster, there is a need for a Mixer output Cluster Descriptor.

The **waClusterDescrID()** array contains the IDs of the Cluster Descriptors that characterize the Clusters that may be exposed on the Output Pin of the Mixer Unit. A value of zero in an array element indicates that the output Cluster is inherited from the current Cluster on Input Pin one (the dominant Input Pin). The **wNrClusterDescrIDs** field contains the number of elements in that array and shall always be greater than zero. For a detailed description of the Cluster Descriptor, see Section 4.4, “Cluster Descriptor”. If the Mixer Unit exposes more than one Cluster Configuration on its Output Pin (**wNrClusterDescrIDs** > 1), then the Cluster Control shall be present. However, the Cluster Active Control shall always be present (and is therefore not part of the **dOptControls** field).

The following table details the structure of the Mixer Unit Descriptor.

Table 4-26: Mixer Unit Descriptor

Offset	Field	Size	Value	Description
0	wLength	2	Number	Size of this Descriptor, in bytes: 22+p*2+q*2.
2	wDescriptorType	2	Constant	EXT_INTERFACE Descriptor type.

Offset	Field	Size	Value	Description
4	wDescriptorSubtype	2	Constant	MIXER_UNIT Descriptor subtype.
6	wDescriptorID	2	Number	Unique ID for this Descriptor.
8	wStrDescriptorID	2	Number	Unique ID for a String Descriptor.
10	wUnitID	2	Number	Value uniquely identifying the Unit within the Audio Function. This value is used in all Commands to address this Unit.
12	dOptControls	4	Bitmap	D0: Cluster Control. D1: Underflow Control. D2: Overflow Control. D31..3: Reserved.
16	wPCC	2	Number	The Pin Channel Count value for the connection that originates at the Output Pin of the Mixer Unit.
18	wNrInputPins	2	Number	Number of Input Pins of this Unit: p
20	waSourceID(1)	2	Number	ID of the Unit or Terminal to which the first Input Pin of this Mixer Unit is connected.
…	…	…	…	…
18+p*2	waSourceID (p)	2	Number	ID of the Unit or Terminal to which the last Input Pin of this Mixer Unit is connected.
20+p*2	wNrClusterDescrIDs	2	Number	Number of elements in the **waClusterDescrID()** array: p.
22+p*2	waClusterDescrID(1)	2	Number	ID of the first Cluster Descriptor for this Mixer Unit.
…	…	…	…	…
20+p*2+q*2	waClusterDescrID(q)	2	Number	ID of the last Cluster Descriptor for this Mixer Unit.

4.5.3.6 SELECTOR UNIT DESCRIPTOR

The Selector Unit is uniquely identified by the value in the **wUnitID** field of the Selector Unit Descriptor. This value shall be passed in the **wEntityID** field of each Command that is directed to the Selector Unit.

The **wNrInputPins** field contains the number of Input Pins (p) of the Selector Unit. The connectivity of the Input Pins is described via the **waSourceID()** array that contains p elements. The index i into the array is one-based and directly related to the Input Pin numbers. **waSourceID(**i**)** contains the ID of the Unit or Terminal to which Input Pin i is connected.

The Selector Unit does not redefine the Cluster on its Output Pin. Rather, it always inherits the Cluster from its currently selected Input Pin. If supported, setting the Selector Control to zero (no connect), switches the output Cluster into the Inactive state.

The Selector Unit Descriptor does not contain a PCC value. The implied PCC value on its Output Pin shall be derived from the maximum of the inherited PCC values on all its Input Pins.

The OCN:ICN:IPN triplet used to access an AudioControl within the Selector Unit shall be set to 0:0:0.

The following table details the structure of the Selector Unit Descriptor.

Table 4-27: Selector Unit Descriptor

Offset	Field	Size	Value	Description
0	wLength	2	Number	Size of this Descriptor, in bytes: 18+p.
2	wDescriptorType	2	Constant	EXT_INTERFACE Descriptor type.
4	wDescriptorSubtype	2	Constant	SELECTOR_UNIT Descriptor subtype.
6	wDescriptorID	2	Number	Unique ID for this Descriptor.
8	wStrDescriptorID	2	Number	Unique ID for a String Descriptor.
10	wUnitID	2	Number	Value uniquely identifying the Unit within the Audio Function. This value is used in all Commands to address this Unit.
12	dOptControls	4	Bitmap	D31..0: Reserved.
16	wNrInputPins	2	Number	Number of Input Pins of this Unit: p
18	waSourceID(1)	2	Number	ID of the Unit or Terminal to which the first Input Pin of this Selector Unit is connected.
…	…	…	…	…
16+p	waSourceID (p)	2	Number	ID of the Unit or Terminal to which the last Input Pin of this Selector Unit is connected.

4.5.3.7 FEATURE UNIT DESCRIPTOR

The Feature Unit is uniquely identified by the value in the **wUnitID** field of the Feature Unit Descriptor. This value shall be passed in the **wEntityID** field of each Command that is directed to the Feature Unit.

The **wSourceID** field is used to describe the connectivity for this Feature Unit. It contains the ID of the Unit or Terminal to which this Feature Unit is connected via its Input Pin. The Cluster Descriptor, describing the n logical channels entering the Feature Unit is not repeated here. It is up to the Host software to trace the connection 'upstream' to locate the Cluster Descriptor pertaining to this Cluster. The Feature Unit does not redefine the output Cluster and thus inherits the output Cluster from the input Cluster.

The Feature Unit Descriptor does not contain a PCC value. The implied PCC value on its Output Pin shall be derived from the inherited PCC value on its Input Pin.

The OCN = ICN used to access an AudioControl within the Feature Unit is the Pin channel number on which the AudioControl resides and is in the range $[0..N]$ where zero represents the Primary channel and N is the number of Pin channels of the Feature Unit ($PCC = N$). Note that if $n < N$, interaction with an AudioControl with an OCN = ICN outside the range $[0..n]$ is still possible, i.e., the accessibility of an AudioControl shall not depend on the current Cluster Configuration. The IPN shall be set to one.

The layout of the Feature Unit Descriptor is detailed in the following table.

Table 4-28: Feature Unit Descriptor

Offset	Field	Size	Value	Description
0	wLength	2	Number	Size of this Descriptor, in bytes: 18+N*4
2	wDescriptorType	2	Constant	EXT_INTERFACE Descriptor type.

Offset	Field	Size	Value	Description
4	wDescriptorSubtype	2	Constant	FEATURE_UNIT Descriptor subtype.
6	wDescriptorID	2	Number	Unique ID for this Descriptor.
8	wStrDescriptorID	2	Number	Unique ID for a String Descriptor.
10	wUnitID	2	Number	Value uniquely identifying the Unit within the Audio Function. This value is used in all Commands to address this Unit.
12	wSourceID	2	Number	ID of the Unit or Terminal to which this Feature Unit is connected.
14	dOptControls(0)	4	Bitmap	The AudioControls bitmap for Primary channel 0: D0: Bypass Control. D1: Mute Control. D2: Gain Control. D3: Bass Control. D4: Mid Control. D5: Treble Control. D6: Graphic Equalizer Control. D7: Automatic Gain Control. D8: Delay Control. D9: Bass Boost Control. D10: Loudness Control. D11: Input Gain Pad Control. D12: Phase Inverter Control. D13: Underflow Control. D14: Overflow Control. D31..15: Reserved.
18	dOptControls(1)	4	Bitmap	The AudioControls bitmap for Pin channel 1.
…	…	…	…	…
14+N*4	dOptControls(N)	4	Bitmap	The AudioControls bitmap for Pin channel N.

4.5.3.8 SAMPLING RATE CONVERTER UNIT DESCRIPTOR

The Sampling Rate Converter Unit Descriptor provides information to the Host that is related to the functional aspects of the SRC Unit.

The SRC Unit is uniquely identified by the value in the **wUnitID** field. This value shall be passed in the **wEntityID** field of each Command that is directed to the Feature Unit.

The **wSourceID** field is used to describe the connectivity for this SRC Unit. It contains the ID of the Unit or Terminal to which this SRC Unit is connected via its Input Pin. The Cluster Descriptor, describing the logical channels entering the SRC Unit is not repeated here. It is up to the Host software to trace the connection ‘upstream’ to locate the Cluster Descriptor pertaining to this Cluster. The SRC Unit does not redefine the output Cluster and thus inherits the output Cluster from the input Cluster.

The **wCSourceInID** field contains the ID of the Clock Entity associated with the audio Input Pin. If the audio Input Pin is not associated with a Clock Entity, then the **wCSourceInID** shall be set to zero**.**

The **wCSourceOutID** field contains the ID of the Clock Entity associated with the audio Output Pin. If the audio Output Pin is not associated with a Clock Entity, then the **wCSourceOutID** shall be set to zero**.**

Note: For the SRC Unit to be useful, at least one of the Clock Input Pins shall be connected to a Clock Entity.

The SRC Unit Descriptor does not contain a PCC value. The implied PCC value on its Output Pin shall be derived from the inherited PCC value on its Input Pin.

The following table presents an outline of the SRC Unit Descriptor.

Table 4-29: Sampling Rate Converter Unit Descriptor

Offset	Field	Size	Value	Description
0	wLength	2	Number	Size of this Descriptor, in bytes: 22.
2	wDescriptorType	2	Constant	EXT_INTERFACE Descriptor type.
4	wDescriptorSubtype	2	Constant	SAMPLE_RATE_CONVERTER Descriptor subtype.
6	wDescriptorID	2	Number	Unique ID for this Descriptor.
8	wStrDescriptorID	2	Number	Unique ID for a String Descriptor.
10	wUnitID	2	Number	Value uniquely identifying the Unit within the Audio Function. This value is used in all Commands to address this Unit.
12	wSourceID	2	Number	ID of the Unit or Terminal to which this SRC Unit is connected.
14	wCSourceInID	2	Number	ID of the Clock Entity to which this SRC Unit input section is connected.
16	wCSourceOutID	2	Number	ID of the Clock Entity to which this SRC Unit output section is connected.
18	dOptControls	4	Bitmap	D0: Underflow Control. D1: Overflow Control. D31..2: Reserved.

4.5.3.9 EFFECT UNIT DESCRIPTOR

The Effect Unit is uniquely identified by the value in the **wUnitID** field of the Effect Unit Descriptor. This value shall be passed in the **wEntityID** field of each Command that is directed to the Effect Unit.

The **wEffectType** field contains a value that fully identifies the Effect Unit. For a list of all supported Effect Unit Types, see Appendix A.19, “Effect Unit Effect Types.”

The **wSourceID** field is used to describe the connectivity for this Effect Unit. It contains the ID of the Unit or Terminal to which this Effect Unit is connected via its Input Pin. The Cluster Descriptor, describing the n logical channels entering the Effect Unit is not repeated here. It is up to the Host software to trace the connection ‘upstream’ to locate the Cluster Descriptor pertaining to this Cluster. The Effect Unit does not redefine the output Cluster and thus inherits the output Cluster from the input Cluster.

The Effect Unit Descriptor does not contain a PCC value. The implied PCC value on its Output Pin shall be derived from the inherited PCC value on its Input Pin.

The OCN = ICN used to access an AudioControl within the Effect Unit is the Pin channel number on which the AudioControl resides and is in the range $[0..N]$ where zero represents the Primary channel and N is the number of Pin channels of the Effect Unit ($PCC = N$). Note that if $n < N$, interaction with an AudioControl with an OCN = ICN outside the range $[0..n]$ is still possible, i.e., the accessibility of an AudioControl shall not depend on the current Cluster Configuration. The IPN shall be set to one.

The following table outlines the Effect Unit Descriptor.

Table 4-30: Effect Unit Descriptor

Offset	Field	Size	Value	Description
0	wLength	2	Number	Size of this Descriptor, in bytes: 20+N*4.
2	wDescriptorType	2	Constant	EXT_INTERFACE Descriptor type.
4	wDescriptorSubtype	2	Constant	EFFECT_UNIT Descriptor subtype.
6	wDescriptorID	2	Number	Unique ID for this Descriptor.
8	wStrDescriptorID	2	Number	Unique ID for a String Descriptor.
10	wUnitID	2	Number	Value uniquely identifying the Unit within the Audio Function. This value is used in all Commands to address this Unit.
12	wEffectType	2	Constant	Constant identifying the type of effect this Unit is performing.
14	wSourceID	2	Number	ID of the Unit or Terminal to which this Effect Unit is connected.
16	daOptControls(0)	4	Bitmap	The AudioControls bitmap for Primary channel 0: D31..0: Effect-specific allocation.
…	…	…	…	…
16+N*4	daOptControls(N)	4	Bitmap	The AudioControls bitmap for Pin channel N: D31..0: Effect-specific allocation.

4.5.3.9.1 PARAMETRIC EQUALIZER SECTION EFFECT UNIT DESCRIPTOR

The **wEffectType** field of the common Effect Unit Descriptor contains the value PARAM_EQ_SECTION_EFFECT. (See Appendix A.19, “Effect Unit Effect Types.”

The following table outlines the PEQS Effect Unit Descriptor. It is identical to the common Effect Unit Descriptor, except for some field values. It is repeated here for clarity.

Table 4-31: Parametric Equalizer Section Effect Unit Descriptor

Offset	Field	Size	Value	Description
0	wLength	2	Number	Size of this Descriptor, in bytes: 20+N*4.
2	wDescriptorType	2	Constant	EXT_INTERFACE Descriptor type.
4	wDescriptorSubtype	2	Constant	EFFECT_UNIT Descriptor subtype.

Offset	Field	Size	Value	Description
6	wDescriptorID	2	Number	Unique ID for this Descriptor.
8	wStrDescriptorID	2	Number	Unique ID for a String Descriptor.
10	wUnitID	2	Number	Value uniquely identifying the Unit within the Audio Function. This value is used in all Commands to address this Unit.
12	wEffectType	2	Constant	PARAM_EQ_SECTION_EFFECT effect type.
14	wSourceID	2	Number	ID of the Unit or Terminal to which this Effect Unit is connected.
16	daOptControls(0)	4	Bitmap	The AudioControls bitmap for Primary channel 0: D0: Bypass Control. D1: Center Frequency Control. D2: Q Factor Control. D3: Gain Control. D4: Underflow Control. D5: Overflow Control. D31..6: Reserved.
…	…	…	…	…
16+N*4	daOptControls(N)	4	Bitmap	The AudioControls bitmap for Pin channel N.

4.5.3.9.2 REVERBERATION EFFECT UNIT DESCRIPTOR

The **wEffectType** field of the common Effect Unit Descriptor contains the value REVERBERATION_EFFECT. See Appendix A.19, “Effect Unit Effect Types.”

The following table outlines the Reverberation Effect Unit Descriptor. It is identical to the common Effect Unit Descriptor, but extended with some additional fields.

The **wNrTypes** field indicates how many different Types of Reverberation the Unit supports. The **wTypeStrDescriptorID()** array points to a human-readable string that describes the Reverberation effect for each of the supported Reverberation Types. The **wTypeStrDescriptorID()** array is optional but when present shall contain an entry for each supported Reverberation Types. The presence of the **wTypeStrDescriptorID()** array shall be derived from the value in the wLength field of the Descriptor.

Table 4-32: Reverberation Effect Unit Descriptor

Offset	Field	Size	Value	Description
0	wLength	2	Number	Size of this Descriptor, in bytes: 22+N*4+n*2.
2	wDescriptorType	2	Constant	EXT_INTERFACE Descriptor type.
4	wDescriptorSubtype	2	Constant	EFFECT_UNIT Descriptor subtype.
6	wDescriptorID	2	Number	Unique ID for this Descriptor.
8	wStrDescriptorID	2	Number	Unique ID for a String Descriptor.

Offset	Field	Size	Value	Description
10	wUnitID	2	Number	Value uniquely identifying the Unit within the Audio Function. This value is used in all Commands to address this Unit.
12	wEffectType	2	Constant	REVERBERATION_EFFECT effect type.
14	wSourceID	2	Number	ID of the Unit or Terminal to which this Effect Unit is connected.
16	daOptControls(0)	4	Bitmap	The AudioControls bitmap for Primary channel 0: D0: Bypass Control. D1: Type Control. D2: Level Control. D3: Time Control. D4: Delay Feedback Control. D5: Pre-Delay Control. D6: Density Control. D7: Hi-Freq Roll-Off Control. D8: Underflow Control. D9: Overflow Control. D31..10: Reserved.
…	…	…	…	…
16+N*4	daOptControls(N)	4	Bitmap	The AudioControls bitmap for Pin channel N.
20+N*4	wNrTypes	2	Number	The number of Reverberation Types supported by this Unit.
22+N*4	wTypeStrDescriptorID(1)	2	Number	Unique ID for a Type String Descriptor.
…	…	…	…	…
20+N*4+n*2	wTypeStrDescriptorID(n)	2	Number	Unique ID for a Type String Descriptor.

4.5.3.9.3 MODULATION DELAY EFFECT UNIT DESCRIPTOR

The **wEffectType** field of the common Effect Unit Descriptor contains the value MOD_DELAY_EFFECT. See Appendix A.19, “Effect Unit Effect Types.”

The following table outlines the Modulation Delay Effect Unit Descriptor. It is identical to the common Effect Unit Descriptor, except for some field values. It is repeated here for clarity.

Table 4-33: Modulation Delay Effect Unit Descriptor

Offset	Field	Size	Value	Description
0	wLength	2	Number	Size of this Descriptor, in bytes: 20+N*4.
2	wDescriptorType	2	Constant	EXT_INTERFACE Descriptor type.
4	wDescriptorSubtype	2	Constant	EFFECT_UNIT Descriptor subtype.
6	wDescriptorID	2	Number	Unique ID for this Descriptor.
8	wStrDescriptorID	2	Number	Unique ID for a String Descriptor.

Offset	Field	Size	Value	Description
10	wUnitID	2	Number	Value uniquely identifying the Unit within the Audio Function. This value is used in all Commands to address this Unit.
12	wEffectType	2	Constant	MOD_DELAY_EFFECT effect type.
14	wSourceID	2	Number	ID of the Unit or Terminal to which this Effect Unit is connected.
16	daOptControls(0)	4	Bitmap	The AudioControls bitmap for Primary channel 0: D0: Bypass Control. D1: Balance Control. D2: Rate Control. D3: Depth Control. D4: Time Control. D5: Feedback Level Control. D6: Underflow Control. D7: Overflow Control. D31..8: Reserved.
…	…	…	…	…
16+N*4	daOptControls(N)	4	Bitmap	The AudioControls bitmap for Pin channel N.

4.5.3.9.4 DYNAMIC RANGE COMPRESSOR/EXPANDER EFFECT UNIT DESCRIPTOR

The **wEffectType** field of the common Effect Unit Descriptor contains the value DYN_RANGE_COMP_EXP_EFFECT. See Appendix A.19, “Effect Unit Effect Types.”

The following table outlines the Dynamic Range Compressor/Expander Effect Unit Descriptor. It is identical to the common Effect Unit Descriptor, except for some field values. It is repeated here for clarity.

Table 4-34: Dynamic Range Compressor/Expander Effect Unit Descriptor

Offset	Field	Size	Value	Description
0	wLength	2	Number	Size of this Descriptor, in bytes: 20+N*4.
2	wDescriptorType	2	Constant	EXT_INTERFACE Descriptor type.
4	wDescriptorSubtype	2	Constant	EFFECT_UNIT Descriptor subtype.
6	wDescriptorID	2	Number	Unique ID for this Descriptor.
8	wStrDescriptorID	2	Number	Unique ID for a String Descriptor.
10	wUnitID	2	Number	Value uniquely identifying the Unit within the Audio Function. This value is used in all Commands to address this Unit.
12	wEffectType	2	Constant	DYN_RANGE_COMP_EXP_EFFECT effect type.
14	wSourceID	2	Number	ID of the Unit or Terminal to which this Effect Unit is connected.

Offset	Field	Size	Value	Description
16	daOptControls(0)	4	Bitmap	The AudioControls bitmap for Primary channel 0: D0: Bypass Control. D1: Compression Ratio Control. D2: MaxAmpl Control. D3: Threshold Control. D4: Attack Time Control. D5: Release Time Control. D6: Underflow Control. D7: Overflow Control. D31..8: Reserved.
…	…	…	…	…
16+N*4	daOptControls(N)	4	Bitmap	The AudioControls bitmap for Pin channel N.

4.5.3.10 PROCESSING UNIT DESCRIPTOR

The Processing Unit is uniquely identified by the value in the **wUnitID** field of the Processing Unit Descriptor. This value shall be passed in the **wEntityID** field of each Command that is directed to the Processing Unit.

The **wProcessType** field contains a value that fully identifies the Processing Unit. For a list of all supported Processing Unit Types, see Appendix A.20, “Processing Unit Process Types.”

The **wNrInputPins** field contains the number of Input Pins (p) of the Processing Unit. The connectivity of the Input Pins is described via the **waSourceID()** array that contains p elements. The index i into the array is one-based and directly related to the Input Pin numbers. **waSourceID(**i**)** contains the ID of the Unit or Terminal to which Input Pin i is connected. The Cluster Descriptors, describing the logical channels entering the Processing Unit are not repeated here. It is up to the Host software to trace the connections ‘upstream’ to locate the Cluster Descriptors pertaining to the Clusters.

The PCC value for each of the Input Pins of the Processing Unit is inherited from the first upstream Entity that defines a PCC value on its Output Pin.

The **wPCC** field contains the Pin Channel Count value for the connection that originates at the Output Pin of the Processing Unit. This also determines the number of Output Pin channels the Processing Unit supports.

Because the Processing Unit can freely redefine the output Cluster Configuration, possibly based on the values of some internal AudioControls, there is a need for an output Cluster Descriptor array.

The **waClusterDescrID()** array contains the IDs of the Cluster Descriptors that characterize the Clusters that may be exposed on the Output Pin of the Processing unit. A value of zero in an array element indicates that the output Cluster is inherited from the current Cluster on Input Pin one (the dominant Input Pin). The **wNrClusterDescrIDs** field contains the number of elements in that array and shall always be greater than zero. For a detailed description of the Cluster Descriptor, see Section 4.4, “Cluster Descriptor”. If the Processing Unit exposes more than one Cluster Configuration on its Output Pin (**wNrClusterDescrIDs** > 1), then the Cluster Control shall be present. However, the Cluster Active Control shall always be present (and is therefore not part of the **dOptControls** field).

The following table outlines the Processing Unit Descriptor.

Table 4-35: Common Part of the Processing Unit Descriptor

Offset	Field	Size	Value	Description
0	wLength	2	Number	Size of this Descriptor, in bytes: 24+p*2+q*2.
2	wDescriptorType	2	Constant	EXT_INTERFACE Descriptor type.
4	wDescriptorSubtype	2	Constant	PROCESSING_UNIT Descriptor subtype.
6	wDescriptorID	2	Number	Unique ID for this Descriptor.
8	wStrDescriptorID	2	Number	Unique ID for a String Descriptor.
10	wUnitID	2	Number	Value uniquely identifying the Unit within the Audio Function. This value is used in all Commands to address this Unit.
12	wProcessType	2	Constant	Constant identifying the type of processing this Unit is performing.
14	dOptControls	4	Bitmap	D0: Bypass Control. D1: Cluster Control. D2: Underflow Control. D3: Overflow Control. D31..4: Process-specific.
18	wPCC	2	Number	The Pin Channel Count value for the connection that originates at the Output Pin of the Processing Unit.
20	wNrInputPins	2	Number	Number of Input Pins of this Unit: p
22	waSourceID(1)	2	Number	ID of the Unit or Terminal to which the first Input Pin of this Processing Unit is connected.
…	…	2	…	…
20+p*2	waSourceID (p)	2	Number	ID of the Unit or Terminal to which the last Input Pin of this Processing Unit is connected.
22+p*2	wNrClusterDescrIDs	2	Number	Number of elements in the **waClusterDescrID()** array: q.
24+p*2	waClusterDescrID(1)	2	Number	Unique ID of the first Cluster Descriptor.
…	…	…	…	…
22+p*2+q*2	waClusterDescrID(q)	2	Number	Unique ID of the last Cluster Descriptor.

4.5.3.10.1 UP/DOWN-MIX PROCESSING UNIT DESCRIPTOR

The **wProcessType** field of the common Processing Unit Descriptor contains the value UP/DOWNMIX_PROCESS. (See Appendix A.20, “Processing Unit Process Types”)

The Cluster Control is used to change the behavior of the Processing Unit by selecting different modes of operation, resulting in a different output Cluster Configuration. If the Up/Down-mix Processing Unit supports more than one mode of operation, this AudioControl shall be present.

The number of supported modes (*n*) is identical to the number of output Clusters the Unit supports and is therefore advertised in the **wNrClusterDescrIDs** field. The index i into this array is one-based and directly related to the number of the mode described by entry **waClusterDescrID(*i*)**. It is the value i that shall be used as a parameter for the Set Cluster Command to select the mode i.

The OCN:ICN:IPN triplet used to access an AudioControl within the Up/Down-mix Processing Unit shall be set to 0:0:0.

Table 4-36: Up/Down-mix Processing Unit Descriptor

Offset	Field	Size	Value	Description
0	wLength	2	Number	Size of this Descriptor, in bytes: 24+p*2+q*2.
2	wDescriptorType	2	Constant	EXT_INTERFACE Descriptor type.
4	wDescriptorSubtype	2	Constant	PROCESSING_UNIT Descriptor subtype.
6	wDescriptorID	2	Number	Unique ID for this Descriptor.
8	wStrDescriptorID	2	Number	Unique ID for a String Descriptor.
10	wUnitID	2	Number	Value uniquely identifying the Unit within the Audio Function. This value is used in all Commands to address this Unit.
12	wProcessType	2	Constant	UP/DOWNMIX_PROCESS process type.
14	dOptControls	4	Bitmap	D0: Bypass Control. D1: Cluster Control. D2: Underflow Control. D3: Overflow Control. D31..4: Reserved.
18	wPCC	2	Number	The Pin Channel Count value for the connection that originates at the Output Pin of the Processing Unit.
20	wNrInputPins	2	Number	Number of Input Pins of this Unit: p
22	waSourceID(1)	2	Number	ID of the Unit or Terminal to which the first Input Pin of this Processing Unit is connected.
…	…	…	…	…
20+p*2	waSourceID (p)	2	Number	ID of the Unit or Terminal to which the last Input Pin of this Processing Unit is connected.
22+p*2	wNrClusterDescrIDs	2	Number	Number of elements in the **waClusterDescrID()** array: q.
24+p*2	waClusterDescrID(1)	2	Number	Unique ID of the first Cluster Descriptor.
…	…	…	…	…
22+p*2+q*2	waClusterDescrID(q)	2	Number	Unique ID of the last Cluster Descriptor.

4.5.3.10.2 CHANNEL REMAP PROCESSING UNIT DESCRIPTOR

The **wProcessType** field of the common Processing Unit Descriptor contains the value CHANNEL_REMAP_PROCESS. (See Appendix A.20, “Processing Unit Process Types”)

The Cluster Control is used to change the behavior of the Processing Unit by selecting different remapping modes, resulting in a different output Cluster Configuration. If the Channel Remap Processing Unit supports more than one remapping mode, this AudioControl shall be present.

The number of supported modes (*n*) is identical to the number of output Clusters the Unit supports and is therefore advertised in the **wNrClusterDescrIDs** field. The index i into this array is one-based and directly related to the number of the mode described by entry **waClusterDescrID(**i**)**. It is the value i that shall be used as a parameter for the Set Cluster Command to select the mode i.

The OCN:ICN:IPN used to access an AudioControl within the Channel Remap Processing Unit shall be set to 0:0:0.

Table 4-37: Channel Remap Processing Unit Descriptor

Offset	Field	Size	Value	Description
0	wLength	2	Number	Size of this Descriptor, in bytes: 24+p*2+q*2.
2	wDescriptorType	2	Constant	EXT_INTERFACE Descriptor type.
4	wDescriptorSubtype	2	Constant	PROCESSING_UNIT Descriptor subtype.
6	wDescriptorID	2	Number	Unique ID for this Descriptor.
8	wStrDescriptorID	2	Number	Unique ID for a String Descriptor.
10	wUnitID	2	Number	Value uniquely identifying the Unit within the Audio Function. This value is used in all Commands to address this Unit.
12	wProcessType	2	Constant	CHANNEL_REMAP_PROCESS process type.
14	dOptControls	4	Bitmap	D0: Bypass Control. D1: Cluster Control. D2: Underflow Control. D3: Overflow Control. D31..4: Reserved.
18	wPCC	2	Number	The Pin Channel Count value for the connection that originates at the Output Pin of the Processing Unit.
20	wNrInputPins	2	Number	Number of Input Pins of this Unit: p
22	waSourceID(1)	2	Number	ID of the Unit or Terminal to which the first Input Pin of this Processing Unit is connected.
…	…	…	…	…
20+p*2	waSourceID (p)	2	Number	ID of the Unit or Terminal to which the last Input Pin of this Processing Unit is connected.
22+p*2	wNrClusterDescrIDs	2	Number	Number of elements in the **waClusterDescrID()** array: q.
24+p*2	waClusterDescrID(1)	2	Number	Unique ID of the first Cluster Descriptor.

Offset	Field	Size	Value	Description
…	…	…	…	…
22+p*2+q*2	waClusterDescrID(q)	2	Number	Unique ID of the last Cluster Descriptor.

4.5.3.10.3 STEREO EXTENDER PROCESSING UNIT DESCRIPTOR

The **wProcessType** field of the common Processing Unit Descriptor contains the value STEREO_EXTENDER_PROCESS. (See Appendix A.20, “Processing Unit Process Types”)

The Stereo Extender Processing Unit has a single Input Pin. Therefore, the **wNrInputs** field shall contain the value 1.

The input Cluster to the Stereo Extender Processing Unit shall contain at least Front Left and Front Right logical input channels. The output Cluster is inherited from the Input Pin and therefore, the **wNrClusterDescrIDs** field shall be set to one and the **waClusterDescrID(1)** field shall be set to zero.

The OCN:ICN:IPN used to access an AudioControl within the Stereo Extender Processing Unit shall be set to 0:0:0.

Table 4-38: Stereo Extender Processing Unit Descriptor

Offset	Field	Size	Value	Description
0	wLength	2	Number	Size of this Descriptor, in bytes: 26.
2	wDescriptorType	2	Constant	EXT_INTERFACE Descriptor type.
4	wDescriptorSubtype	2	Constant	PROCESSING_UNIT Descriptor subtype.
6	wDescriptorID	2	Number	Unique ID for this Descriptor.
8	wStrDescriptorID	2	Number	Unique ID for a String Descriptor.
10	wUnitID	2	Number	Value uniquely identifying the Unit within the Audio Function. This value is used in all Commands to address this Unit.
12	wProcessType	2	Constant	STEREO_EXTENDER_PROCESS process type.
14	dOptControls	4	Bitmap	D0: Bypass Control. D1: Cluster Control not Implemented. Shall be set to zero. D2: Underflow Control. D3: Overflow Control. D31..4: Reserved.
18	wPCC	2	Number	The Pin Channel Count value for the connection that originates at the Output Pin of the Processing Unit.
20	wNrInputPins	2	Number	Number of Input Pins of this Unit: 1
22	waSourceID(1)	2	Number	ID of the Unit or Terminal to which the Input Pin of this Processing Unit is connected.
24	wNrClusterDescrIDs	2	Number	Number of elements in the **waClusterDescrID()** array: 1.
26	waClusterDescrID(1)	2	Number	Shall be set to zero.

4.5.3.10.4 MULTI-FUNCTION PROCESSING UNIT DESCRIPTOR

The **wProcessType** field of the Processing Unit Descriptor contains the value MULTI_FUNCTION_PROCESS. (See Appendix A.20, “Processing Unit Process Types”)

The Multi-Function Processing Unit may have multiple Input Pins as indicated in the **wNrInputPins** field.

The OCN:ICN:IPN used to access an AudioControl within the Multi-Function Processing Unit shall be set to 0:0:0.

The Multi-Function Processing Unit can redefine its output Cluster Configurations, depending on which algorithms are currently active. The Read-Only (r) Cluster Control is used to advertise the current output Cluster Configuration.

The Bypass Control shall be implemented.

The mandatory Read-Only (r) Algo Present Control returns a bitmap indicating what types of algorithms are performed inside the Multi-Function Processing Unit. Multiple bits may be set simultaneously.

The optional Algo Enable Control provides a means to selectively enable or disable the implemented algorithms.

The following table outlines the Multi-Function Processing Unit Descriptor.

Table 4-39: Multi-Function Processing Unit Descriptor

Offset	Field	Size	Value	Description
0	wLength	2	Number	Size of this Descriptor, in bytes: 24+p*2+q*2.
2	wDescriptorType	2	Constant	EXT_INTERFACE Descriptor type.
4	wDescriptorSubtype	2	Constant	PROCESSING_UNIT Descriptor subtype.
6	wDescriptorID	2	Number	Unique ID for this Descriptor.
8	wStrDescriptorID	2	Number	Unique ID for a String Descriptor.
10	wUnitID	2	Number	Value uniquely identifying the Unit within the Audio Function. This value is used in all Commands to address this Unit.
12	wProcessType	2	Constant	MULTI_FUNCTION_PROCESS process type.
14	dOptControls	4	Bitmap	D0: Cluster Control. D1: Algo Enable Control. D2: Underflow Control. D3: Overflow Control. D31..4: Reserved.
18	wPCC	2	Number	The Pin Channel Count value for the connection that originates at the Output Pin of the Processing Unit.
20	wNrInputPins	2	Number	Number of Input Pins of this Unit: p
22	waSourceID(1)	2	Number	ID of the Unit or Terminal to which the first Input Pin of this Processing Unit is connected.
…	…	…	…	…

Offset	Field	Size	Value	Description
20+p*2	waSourceID (p)	2	Number	ID of the Unit or Terminal to which the last Input Pin of this Processing Unit is connected.
22+p*2	wNrClusterDescrIDs	2	Number	Number of elements in the **waClusterDescrID()** array: q.
24+p*2	waClusterDescrID(1)	2	Number	Unique ID of the first Cluster Descriptor.
…	…	…	…	…
22+p*2+q*2	waClusterDescrID(q)	2	Number	Unique ID of the last Cluster Descriptor.

4.5.3.11 EXTENSION UNIT DESCRIPTOR

The Extension Unit is uniquely identified by the value in the **wUnitID** field of the Extension Unit Descriptor. This value shall be passed in the **wEntityID** field of each Command that is directed to the Extension Unit.

The Extension Unit Descriptor provides just enough information about the Extension Unit so that a generic Audio Class driver can be aware of vendor-specific components within the Audio Function. The **guidExtensionCode** field shall contain a vendor-specific code in the form of a GUID that further identifies the Extension Unit. Note that the GUID is used to uniquely identify the functionality and behavior of the Extension Unit. Therefore, the same GUID shall be used in all implementations that expose the Extension Unit with this functionality and behavior.

For more information about the generation and use of a GUID, see [IETF RFC 4122 GUID].

The **wNrInputPins** field contains the number of Input Pins (*p*) of the Extension Unit. The connectivity of the Input Pins is described via the **waSourceID()** array that contains *p* elements. The index i into the array is one-based and directly related to the Input Pin numbers. **waSourceID(**i**)** contains the ID of the Unit or Terminal to which Input Pin i is connected. The Cluster Descriptors that describe the logical channels that enter the Extension Unit are not repeated here. It is up to the Host software to trace the connections 'upstream' to locate the Cluster Descriptors pertaining to the Clusters.

The **waPCC()** array contains the PCC values N_p the Extension Unit supports on each of its Input Pins, excluding the Primary channel. However, the actual number of logical channels used on each Input Pin at any given time is solely determined by the number of channels in the incoming Clusters on each Input Pin.

The Extension Unit can freely redefine its output Cluster Configurations, based on internal settings of the Extension Unit's vendor-defined AudioControls.

The Bypass Control shall be implemented.

The **waClusterDescrID()** array contains the IDs of the Cluster Descriptors that characterize the Clusters that may be exposed on the Output Pin of the Extension Unit. A value of zero in an array element indicates that the output Cluster is inherited from the current Cluster on Input Pin one (the dominant Input Pin). The **wNrClusterDescrIDs** field contains the number of elements in that array and shall always be greater than zero. For a detailed description of the Cluster Descriptor, see Section 4.4, "Cluster Descriptor". If the Extension Unit exposes more than one Cluster Configuration on its Output Pin (**wNrClusterDescrIDs** > 1), then the Cluster Control shall be present. However, the Cluster Active Control shall always be present (and is therefore not part of the **dOptControls** field).

The OCN:ICN:IPN used to access a class-defined AudioControl within the Extension Unit shall be set to 0:0:0.

The following table outlines the Extension Unit Descriptor.

Table 4-40: Extension Unit Descriptor

Offset	Field	Size	Value	Description
0	wLength	2	Number	Size of this Descriptor, in bytes: 38+p*2+q*2.
2	wDescriptorType	2	Constant	EXT_INTERFACE Descriptor type.
4	wDescriptorSubtype	2	Constant	EXTENSION_UNIT Descriptor subtype.
6	wDescriptorID	2	Number	Unique ID for this Descriptor.
8	wStrDescriptorID	2	Number	Unique ID for a String Descriptor.
10	wUnitID	2	Number	Value uniquely identifying the Unit within the Audio Function. This value is used in all Commands to address this Unit.
12	guidExtensionCode	16	GUID	Vendor-specific code identifying the Extension Unit.
28	dOptControls	4	Bitmap	D0: Cluster Control. D1: Underflow Control. D2: Overflow Control. D31..3: Reserved.
32	wPCC	2	Number	The Pin Channel Count value for the connection that originates at the Output Pin of the Processing Unit.
34	wNrInputPins	2	Number	Number of Input Pins of this Unit: p
36	waSourceID(1)	2	Number	ID of the Unit or Terminal to which the first Input Pin of this Processing Unit is connected.
…	…	…	…	…
34+p*2	waSourceID (p)	2	Number	ID of the Unit or Terminal to which the last Input Pin of this Processing Unit is connected.
36+p*2	wNrClusterDescrIDs	2	Number	Number of elements in the **waClusterDescrID()** array: q.
38+p*2	waClusterDescrID(1)	2	Number	Unique ID of the first Cluster Descriptor.
…	…	…	…	…
36+p*2+q*2	waClusterDescrID(q)	2	Number	Unique ID of the last Cluster Descriptor.

4.5.3.12 CLOCK SOURCE DESCRIPTOR

The Clock Source Entity is uniquely identified by the value in the **wClockID** field of the Clock Source Entity Descriptor. This value shall be passed in the **wEntityID** field of each Command that is directed to the Clock Source Entity.

The Clock Frequency Control shall always be present and may be implemented as either Read (r) or Read-Write (rw). The supported clock frequencies can be derived from the Range Attribute of the Clock Frequency Control (fixed rate vs. variable rate). Note that even a Clock Source of Type External may be implemented as Read-Write (rw) if the Audio Function can influence that external clock through means outside of USB. The actual sampling

frequency of the Clock Source can be manipulated through the Clock Frequency Command. In addition, the Clock Source can be queried for the validity of its current sampling clock signal through a Get Clock Valid Command.

The **wAttributes** field contains a Clock Type bit field (D0) that indicates whether the Clock Source represents an external clock (D0 = 0b0) or an internal clock (D0=0b1).

The **wClockDomainID** field contains the ID of the Clock Domain from which this Clock Source Entity derives its reference clock.

A value of zero indicates that this Clock Source Entity is independent and free running. Multiple Clock Source Entities may have a value of zero in this field, which means that they are all independent of one another. (It does not mean that these Clock Source Entities belong to the same Clock Domain with ID zero.)

A value of 0xFFFF indicates that the Clock Source Entity is synchronized to SOF.

Any other value indicates the ID of the Clock Domain to which this Clock Source Entity is synchronized.

The **wReferenceTerminal** field contains a reference to a Terminal from which the Clock Source is derived. This is useful for instance when a Clock Source's clock signal is derived from the input signal on an S/PDIF connector, which is represented by an Input Terminal. If the Clock Source is free running or derived from USB SOF (not derived from a Terminal), this field shall be set to zero.

The OCN:ICN:IPN used to access an AudioControl within the Clock Source Entity shall be set to 0:0:0.

The following table outlines the Clock Source Descriptor.

Table 4-41: Clock Source Descriptor

Offset	Field	Size	Value	Description
0	wLength	2	Number	Size of this Descriptor, in bytes: 22.
2	wDescriptorType	2	Constant	EXT_INTERFACE Descriptor type.
4	wDescriptorSubtype	2	Constant	CLOCK_SOURCE Descriptor subtype.
6	wDescriptorID	2	Number	Unique ID for this Descriptor.
8	wStrDescriptorID	2	Number	Unique ID for a String Descriptor.
10	wClockID	2	Number	Value uniquely identifying the Clock Source Entity within the Audio Function. This value is used in all Commands to address this Entity.
12	wAttributes	2	Bitmap	D0: Clock Type: 0: External Clock. 1: Internal Clock. D15..1: Reserved.
14	wClockDomainID	2	Number	0x0000: Independent. 0x0001..0xFFFE: ID of the reference Clock Domain. 0xFFFF: Synchronized to SOF.
16	dOptControls	4	Bitmap	D31..0: Reserved.
20	wReferenceTerminal	2	Number	Terminal ID of the Terminal from which this Clock Source is derived.

4.5.3.13 CLOCK SELECTOR DESCRIPTOR

The Clock Selector Entity is uniquely identified by the value in the **wClockID** field of the Clock Selector Entity Descriptor. This value shall be passed in the **wEntityID** field of each Command that is directed to the Clock Selector Entity.

The **wNrInputPins** field contains the number of Clock Input Pins (p) of the Clock Selector Entity. The connectivity of the Input Pins is described via the **waCSourceID()** array that contains p elements. The index i into the array is one-based and directly related to the Clock Input Pin numbers. **waCSourceID(i)** contains the ID of the Clock Entity to which Clock Input Pin i is connected.

The OCN:ICN:IPN used to access an AudioControl within the Clock Selector Entity shall be set to 0:0:0.

The following table outlines the Clock Selector Descriptor.

Table 4-42: Clock Selector Descriptor

Offset	Field	Size	Value	Description
0	wLength	2	Number	Size of this Descriptor, in bytes: 18+p*2.
2	wDescriptorType	2	Constant	EXT_INTERFACE Descriptor type.
4	wDescriptorSubtype	2	Constant	CLOCK_SELECTOR Descriptor subtype.
6	wDescriptorID	2	Number	Unique ID for this Descriptor.
8	wStrDescriptorID	2	Number	Unique ID for a String Descriptor.
10	wClockID	2	Number	Value uniquely identifying the Clock Selector Entity within the Audio Function. This value is used in all Commands to address this Entity.
12	dOptControls	4	Bitmap	D31..0: Reserved.
16	wNrInputPins	2	Number	Number of Input Pins of this Unit: p.
18	waCSourceID(1)	2	Number	ID of the Clock Entity to which the first Clock Input Pin of this Clock Selector Entity is connected.
…	…	…	…	…
16+p*2	waCSourceID (p)	2	Number	ID of the Clock Entity to which the last Clock Input Pin of this Clock Selector Entity is connected.

4.5.3.14 CONNECTOR ENTITY DESCRIPTOR

The Connector Entity Descriptor returns information about the Attributes of a physical Connector, located on the enclosure that houses the Audio Function. This Connector is typically user accessible.

The **wConID** field contains a unique identifier for the Connector. The primary use for this is to indicate that the same Connector is associated with multiple Terminals. For example, one headset Connector may incorporate the signals for the stereo headphone of the headset and the signal for the mono microphone. This Connector would therefore be part of the Output Terminal that represents the stereo headphone but also be part of the Input Terminal that represents the microphone. This Connector would then be listed in the VARIANT_ENTITIES part of both the Input Terminal and Output Terminal Descriptor, using the same **wConID** value in both Descriptors to indicate the binding.

The **wConType** field contains a value that identifies the physical appearance of the Connector. The constant definitions for the **wConType** field can be found in Appendix A.24, “Connector Types”.

The **wConAttributes** field contains a bitmap that identifies the gender of the Connector (D1..0).

The **dConColor** field contains either 0x00 in the upper byte and the RGB-coded color of the Connector in the lower 3 bytes or 0x01 in the upper byte and 0x000000 in the lower 3 bytes to indicate color unspecified.

The OCN:ICN:IPN used to access an AudioControl within the Connector Entity shall be set to 0:0:0.

The following table outlines the Connector Entity Descriptor.

Table 4-43: Connector Entity Descriptor

Offset	Field	Size	Value	Description
0	wLength	2	Number	Size of this Descriptor, in bytes: 24.
2	wDescriptorType	2	Constant	EXT_INTERFACE Descriptor type.
4	wDescriptorSubtype	2	Constant	CONNECTOR Descriptor subtype.
6	wDescriptorID	2	Number	Unique ID of this Connector Descriptor.
8	wStrDescriptorID	2	Number	Unique ID for a String Descriptor.
10	wConID	2	Number	Unique ID for the Connector Entity. Can be used to indicate that the same Connector is associated with multiple Terminals.
12	dOptControls	4	Bitmap	D0: Insertion Control. D31..1: Reserved.
16	wConType	2	Constant	The Connector Type of the Connector.
18	wConAttributes	2	Bitmap	D1..0: Gender: 00: Gender Neutral. 01: Plug. 10: Receptacle. 11: Reserved. D7..2: Reserved.
20	dConColor	4	Number	The Connector color of the Connector.

4.5.3.15 POWER DOMAIN ENTITY DESCRIPTOR

Power Domain Entity Descriptors provide information to the Host regarding the existence of one or more Power Domains within the Audio Function and list the Entities, through their respective Entity ID, that are explicit members of a particular Power Domain.

There is a Power Domain Entity Descriptor for each Power Domain in the Audio Function. Therefore, the number of Power Domain Entity Descriptors is an indicator for the number of separately managed Power Domains in the Audio Function.

The Power Domain and its associated Power Domain Entity are uniquely identified by the value in the **wPowerDomainID** field of the Power Domain Entity Descriptor. This value shall be passed in the **wEntityID** field of each Command that is directed to the Power Domain Entity.

The **wEntryTime1..4** fields contain the approximate entry time from Power State PS0 to Power States PS1, PS2, PS3, and PS4, respectively.

The **wExitTime1..4 fields** contain the approximate exit time from Power State PS1, PS2, PS3, and PS4 to Power State PS0, respectively.

The **waEntityID()** array contains the Entity IDs of the explicit Member Entities. The **wNrEntityIDs** field contains the number of elements in that array.

The OCN:ICN:IPN used to access an AudioControl within the Power Domain Entity shall be set to 0:0:0.

The **wStrDescriptorID** field provides the ID of a String Descriptor to further describe the Power Domain Entity.

Table 4-44: Power Domain Entity Descriptor

Offset	Field	Size	Value	Description
0	wLength	2	Number	Size of this Descriptor, in bytes: 34+p*2.
2	wDescriptorType	2	Constant	EXT_INTERFACE Descriptor type.
4	wDescriptorSubtype	2	Constant	POWER_DOMAIN Descriptor subtype.
6	wDescriptorID	2	Number	Unique ID for this Descriptor.
8	wStrDescriptorID	2	Number	Unique ID for a String Descriptor.
10	wPowerDomainID	2	Number	Value uniquely identifying the Power Domain Entity within the Audio Function. This value is used in all Commands to address this Power Domain Entity.
12	dOptControls	4	Bitmap	D31..0: Reserved.
16	wEntryTime1	2	Number	Time to enter from PS0 to PS1. Expressed in 50 µs increments.
18	wExitTime1	2	Number	Time to exit from PS1 to PS0. Expressed in 50 µs increments.
20	wEntryTime2	2	Number	Time to enter from PS0 to PS2. Expressed in 50 µs increments.
22	wExitTime2	2	Number	Time to exit from PS2 to PS0. Expressed in 50 µs increments.
24	wEntryTime3	2	Number	Time to enter from PS0 to PS3. Expressed in 50 µs increments.
26	wExitTime3	2	Number	Time to exit from PS3 to PS0. Expressed in 50 µs increments.
28	wEntryTime4	2	Number	Time to enter from PS0 to PS4. Expressed in 50 µs increments.
30	wExitTime4	2	Number	Time to exit from PS4 to PS0. Expressed in 50 µs increments.
32	wNrEntityIDs	2	Number	Number of elements in the **baEntityID()** array: p.
34	waEntityID(1)	2	Number	ID of the first Entity that belongs to this Power Domain.
…	…	…	…	

Offset	Field	Size	Value	Description
32+p*2	waEntityID(p)	2	Number	ID of the last Entity that belongs to this Power Domain.

4.5.3.16 ENTITYGROUP DESCRIPTOR

The **waEntityID()** array contains the Entity IDs of the Members of the EntityGroup. The **wNrEntityIDs** field contains the number of elements in that array.

Table 4-45: EntityGroup Descriptor

Offset	Field	Size	Value	Description
0	wLength	2	Number	Size of this Descriptor, in bytes: 12+p*2.
2	wDescriptorType	2	Constant	EXT_INTERFACE Descriptor type.
4	wDescriptorSubtype	2	Constant	ENTITY_GROUP Descriptor subtype.
6	wDescriptorID	2	Number	Unique ID for this Descriptor.
8	wStrDescriptorID	2	Number	Unique ID for a String Descriptor.
10	wNrEntityIDs	2	Number	Number of elements in the **baEntityID()** array: p.
12	waEntityID(1)	2	Number	ID of the first Entity that belongs to this EntityGroup.
…	…	…	…	
10+p*2	waEntityID(p)	2	Number	ID of the last Entity that belongs to this EntityGroup.

4.5.3.17 COMMITGROUP DESCRIPTOR

The **waAudioControl()** array contains all the information necessary to fully identify each AudioControl Member of the CommitGroup. The **wNrAudioControls** field contains the number of AudioControls in the array.

Table 4-46: CommitGroup Descriptor

Offset	Field	Size	Value	Description
0	wLength	2	Number	Size of this Descriptor, in bytes: 12+p*10.
2	wDescriptorType	2	Constant	EXT_INTERFACE Descriptor type.
4	wDescriptorSubtype	2	Constant	COMMIT_GROUP Descriptor subtype.
6	wDescriptorID	2	Number	Unique ID for this Descriptor.
8	wStrDescriptorID	2	Number	Unique ID for a String Descriptor.
10	wNrAudioControls	2	Number	Number of AudioControls in the Commitgroup: p.
12	waAudioControl(1).CS	2	Number	Control Selector CS of the first AudioControl in the CommitGroup.
14	waAudioControl(1).Entity	2	Number	ID of the Entity to which the first AudioControl in the CommitGroup belongs.
16	waAudioControl(1).OCN	2	Number	OCN of the first AudioControl in the CommitGroup.

Offset	Field	Size	Value	Description
18	waAudioControl(1).ICN	2	Number	ICN of the first AudioControl in the CommitGroup.
20	waAudioControl(1).IPN	2	Number	IPN of the first AudioControl in the CommitGroup.
…	…	…	…	
12+(p-1)*10	waAudioControl(p).CS	2	Number	Control Selector CS of the last AudioControl in the CommitGroup.
14+(p-1)*10	waAudioControl(p).Entity	2	Number	ID of the Entity to which the last AudioControl in the CommitGroup belongs.
16+(p-1)*10	waAudioControl(p).OCN	2	Number	OCN of the last AudioControl in the CommitGroup.
18+(p-1)*10	waAudioControl(p).ICN	2	Number	ICN of the last AudioControl in the CommitGroup.
20+(p-1)*10	waAudioControl(p).IPN	2	Number	IPN of the last AudioControl in the CommitGroup.

4.6 AUDIOCONTROL ENDPOINT DESCRIPTORS

The following sections describe all possible Endpoint-related Descriptors for the AudioControl Interface.

4.6.1 AUDIOCONTROL CONTROL ENDPOINT DESCRIPTORS

4.6.1.1 STANDARD AUDIOCONTROL CONTROL ENDPOINT DESCRIPTOR

The AudioControl Interface uses the default Endpoint 0. Therefore, there is no dedicated standard AudioControl Control Endpoint Descriptor.

4.6.1.2 CLASS-SPECIFIC AUDIOCONTROL CONTROL ENDPOINT DESCRIPTOR

There is no dedicated class-specific AudioControl Control Endpoint Descriptor.

4.6.2 AUDIOCONTROL INTERRUPT ENDPOINT DESCRIPTORS

4.6.2.1 STANDARD AUDIOCONTROL INTERRUPT ENDPOINT DESCRIPTOR

The interrupt Endpoint Descriptor is identical to the standard Endpoint Descriptor defined in the *USB Core Specifications*. Its fields are set to reflect the interrupt type of the Endpoint. This Endpoint is optional.

The following table outlines the standard AudioControl Interrupt Endpoint Descriptor.

Table 4-47: Standard AudioControl Interrupt Endpoint Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	Number	Size of this Descriptor, in bytes: 7
1	bDescriptorType	1	Constant	ENDPOINT Descriptor type

Offset	Field	Size	Value	Description
2	bEndpointAddress	1	Endpoint	The address of the Endpoint on the USB Device described by this Descriptor. The address is encoded as follows: D7: Direction. 1 = IN Endpoint. D6..4: Reserved. D3..0: The Endpoint number, determined by the designer.
3	bmAttributes	1	Bitmap	D1..0: Transfer Type 11 = Interrupt. All other bits are reserved.
4	wMaxPacketSize	2	Number	Maximum packet size this Endpoint is capable of sending or receiving when this configuration is selected. Used here to pass 6-byte Interrupt Data Message.
6	bInterval	1	Number	Interval for polling the Interrupt Endpoint.

4.6.2.2 CLASS-SPECIFIC AUDIOCONTROL INTERRUPT ENDPOINT DESCRIPTOR

There is no class-specific AudioControl interrupt Endpoint Descriptor.

4.7 AUDIOSTREAMING INTERFACE DESCRIPTORS

The AudioStreaming (AS) Interface Descriptors contain all relevant information to characterize the AudioStreaming Interface in full.

4.7.1 STANDARD AUDIOSTREAMING INTERFACE DESCRIPTOR

The standard AudioStreaming Interface Descriptor is identical to the standard interface Descriptor defined in the *USB Core Specifications*, except that some fields now have dedicated values.

Table 4-48: Standard AudioStreaming Interface Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	Number	Size of this Descriptor, in bytes: 9.
1	bDescriptorType	1	Constant	INTERFACE Descriptor type.
2	bInterfaceNumber	1	Number	Number of the interface. A zero-based value identifying the index in the array of concurrent interfaces supported by this configuration.
3	bAlternateSetting	1	Number	Value used to select an Alternate Setting for the interface identified in the prior field.
4	bNumEndpoints	1	Number	Number of Endpoints used by this interface (excluding Endpoint 0). Shall be either 0 (no data Endpoint), 1 (data Endpoint) or 2 (data and explicit feedback Endpoint).
5	bInterfaceClass	1	AUDIO	Audio Interface Class code (assigned by the USB). See Appendix A.5, “Audio Interface Class Code.”

Offset	Field	Size	Value	Description
6	bInterfaceSubClass	1	AUDIO STREAMING	Audio Interface Subclass code. Assigned by this specification. See Appendix A.6, “Audio Interface Subclass Codes.”
7	bInterfaceProtocol	1	IP_VERSION_04_00	Interface Protocol code. Indicates the current version of the specification. See Appendix A.7, “Audio Interface Protocol Codes.”
8	iInterface	1	Index	Index of a String Descriptor that describes this interface.

4.7.2 CLASS-SPECIFIC AUDIOSTREAMING INTERFACE DESCRIPTOR

The class-specific AudioStreaming Interface Descriptor is a traditional Descriptor that contains the list of Extended Descriptor IDs in the **waDescriptorID()** array, used to fully describe the AudioStreaming Interface.

The order in which the Descriptor IDs are reported is not important because every Descriptor can be identified through its **bDescriptorType** and **bDescriptorSubtype** field.

The following table defines the class-specific AudioStreaming Interface Descriptor.

Table 4-49: Class-specific AudioStreaming Interface Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	Number	Size of this Descriptor, in bytes: 4+n*2.
1	bDescriptorType	1	Constant	CS_INTERFACE Descriptor type.
2	bDescriptorSubtype	1	Constant	AS_GENERIC Descriptor subtype.
3	bNrDescriptorIDs	1	Number	Number of Descriptor IDs in the list: n.
4	waDescriptorID(1)	2	Bitmap	First Descriptor ID.
…	…	…	…	…
4+(n-1)*2	waDescriptorID(n)	2	Bitmap	Last Descriptor ID.

If there is a need to list more than 125 Descriptor IDs, then more Descriptors of subtype AS_GENERIC may be included.

4.7.3 CLASS-SPECIFIC AUDIOSTREAMING SELF DESCRIPTOR

The fields in this Descriptor are described in detail in Section 7.4, “Class-specific AudioStreaming Self Descriptor.”

Table 4-50: Class-specific AudioStreaming Self Descriptor

Offset	Field	Size	Value	Description
0	wLength	2	Number	Size of this Descriptor in bytes: 28.
2	wDescriptorType	2	Constant	EXT_INTERFACE Descriptor type.
4	wDescriptorSubtype	2	Constant	AS_SELF Descriptor subtype.
6	wDescriptorID	2	Number	Unique ID for this Descriptor.
8	wStrDescriptorID	2	Number	Unique ID for a String Descriptor.

Offset	Field	Size	Value	Description
10	dOptControls	4	Bitmap	D0: Active Alternate Setting Control. D1: Valid Alternate Settings Control. D31..2: Reserved.
14	wStartDelayUnits	2	Number	Indicates the units used for the wStartDelay field: 0: Undefined. 1: Milliseconds. 2: Decoded PCM samples. All other values are Reserved.
16	wStartDelay	2	Number	Indicates the time it takes this Interface to reliably produce or consume streamed data. Units used depend on the value of the bStartDelayUnits field.
18	wFormat	2	Number	The Audio Data Format used to communicate with this Alternate Setting of the AudioStreaming Interface. See Section 7, “Audio Data Formats” for further details.
20	wSubslotSize	2	Number	The number of bytes occupied by one audio subslot.
22	wBitResolution	2	Number	The number of effectively used bits from the available bits in an audio subslot.
24	wAuxProtocols	2	Bitmap	Bitmap, indicating which Auxiliary Protocols are required.
26	wControlSize	2	Number	Size of the Control Channel Words, in bytes.

4.7.4 CLASS-SPECIFIC AUDIOSTREAMING VALID FREQUENCY RANGE DESCRIPTOR

The AudioStreaming Valid Frequency Range Descriptor provides information to the Host about what sampling frequency ranges are supported by this Alternate Setting of the AudioStreaming Interface. An Audio Function shall provide this Descriptor to indicate that this Alternate Setting of the interface is only valid if the selected sampling frequency is in the range [dMin..dMax]. The values of dMin and dMax are expressed in Hz. If the Alternate Setting of the interface is valid for any available clock frequency supported by the Audio Function, the Descriptor may be omitted. Multiple instances of this Descriptor are allowed in the same Alternate Setting of the AudioStreaming Interface to describe disjoint frequency ranges.

Table 4-51: Class-specific AudioStreaming Valid Frequency Range Descriptor

Offset	Field	Size	Value	Description
0	wLength	2	Number	Size of this Descriptor in bytes: 18.
2	wDescriptorType	2	Constant	EXT_INTERFACE Descriptor type.
4	wDescriptorSubtype	2	Constant	AS_VALID_FREQ_RANGE Descriptor subtype.
6	wDescriptorID	2	Number	Unique ID for this Descriptor.
8	wStrDescriptorID	2	Number	Unique ID for a String Descriptor.

Offset	Field	Size	Value	Description
10	dMin	4	Number	The minimum sampling frequency at which this Alternate Setting of the AudioStreaming Interface is valid.
14	dMax	4	Number	The maximum sampling frequency at which this Alternate Setting of the AudioStreaming Interface is valid.

4.8 AUDIOSTREAMING ENDPOINT DESCRIPTORS

The following sections describe all possible Endpoint-related Descriptors for the AudioStreaming Interface.

4.8.1 AUDIOSTREAMING ISOCHRONOUS AUDIO DATA ENDPOINT DESCRIPTORS

The standard and class-specific audio data Endpoint Descriptors provide pertinent information on how audio data streams are communicated to the Audio Function. In addition, specific Endpoint capabilities and properties are reported.

4.8.1.1 STANDARD AUDIOSTREAMING ISOCHRONOUS AUDIO DATA ENDPOINT DESCRIPTOR

The standard AudioStreaming isochronous audio data Endpoint Descriptor is identical to the standard Endpoint Descriptor defined in the *USB Core Specifications*. D7 of the **bEndpointAddress** field indicates whether the Endpoint is an audio source (D7 = 0b1) or an audio sink (D7 = 0b0). The **bmAttributes** Field bits are set to reflect the isochronous type of the Endpoint. The synchronization type is indicated by D3..2 and shall be set to Asynchronous, Adaptive or Synchronous. For further details, refer to the *USB Core Specifications*.

Table 4-52: Standard AudioStreaming Isochronous Audio Data Endpoint Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	Number	Size of this Descriptor, in bytes: 7
1	bDescriptorType	1	Constant	ENDPOINT Descriptor type
2	bEndpointAddress	1	Endpoint	The address of the Endpoint on the USB Device described by this Descriptor. The address is encoded as follows: D3..0: The Endpoint number, determined by the designer. D6..4: Reserved. D7: Direction: 0 = OUT Endpoint. 1 = IN Endpoint.

Offset	Field	Size	Value	Description
3	bmAttributes	1	Bitmap	D1..0: Transfer type: 01 = Isochronous. D3..2: Synchronization Type: 01 = Asynchronous. 10 = Adaptive. 11 = Synchronous. D5..4: Usage Type: 00 = Data Endpoint or 10 = Implicit feedback Data Endpoint. All other bits are reserved.
4	wMaxPacketSize	2	Number	(Speed-dependent.) Indicates the maximum packet size and potentially any bursting on this Endpoint. This is determined by the audio bandwidth constraints of the Endpoint.
6	bInterval	1	Number	Interval for polling Endpoint for data transfers.

Note: For SuperSpeed and SuperSpeedPlus Endpoints, the SuperSpeed Endpoint Companion and SuperSpeedPlus Endpoint Companion Descriptors would follow the standard Endpoint Descriptor. See the *USB 3.1 specification* for details.

4.8.1.2 CLASS-SPECIFIC AUDIOSTREAMING ISOCHRONOUS AUDIO DATA ENDPOINT DESCRIPTOR

There is no class-specific AudioStreaming isochronous Audio Data Endpoint Descriptor.

4.8.2 AUDIOSTREAMING ISOCHRONOUS FEEDBACK ENDPOINT DESCRIPTOR

This Descriptor is present only when one or more isochronous audio data Endpoints of the adaptive source type or the asynchronous sink type are implemented.

4.8.2.1 STANDARD AUDIOSTREAMING ISOCHRONOUS FEEDBACK ENDPOINT DESCRIPTOR

The isochronous feedback Endpoint Descriptor is identical to the standard Endpoint Descriptor defined in the *USB Core Specifications*. The **bmAttributes** field bits are set to reflect the isochronous type and synchronization type of the Endpoint.

Table 4-53: Standard AudioStreaming Isochronous Feedback Endpoint Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	Number	Size of this Descriptor, in bytes: 7
1	bDescriptorType	1	Constant	ENDPOINT Descriptor type.

Offset	Field	Size	Value	Description
2	bEndpointAddress	1	Endpoint	The address of the Endpoint on the USB Device described by this Descriptor. The address is encoded as follows: D3..0: The Endpoint number, determined by the designer. D6..4: Reserved. D7: Direction: 0 = OUT Endpoint. 1 = IN Endpoint.
3	bmAttributes	1	Bitmap	D1..0: Transfer type: 01 = Isochronous. D3..2: Synchronization Type: 00 = No Synchronization. D5..4: Usage Type: 01 = Feedback Endpoint. All other bits are reserved.
4	wMaxPacketSize	2	Number	Maximum packet size this Endpoint is capable of sending or receiving when this configuration is selected.
6	bInterval	1	Number	Interval for polling Endpoint for data transfers.

Note: For SuperSpeed and SuperSpeedPlus Endpoints, the SuperSpeed Endpoint Companion and SuperSpeedPlus Endpoint Companion Descriptors would follow the standard Endpoint Descriptor. See the *USB 3.1 specification* for details.

4.8.2.2 CLASS-SPECIFIC AUDIOSTREAMING ISOCHRONOUS FEEDBACK ENDPOINT DESCRIPTOR

There is no class-specific AudioStreaming isochronous feedback Endpoint Descriptor.

4.9 CLASS-SPECIFIC STRING DESCRIPTORS

This specification defines a new type of class-specific String Descriptor. Class-specific String Descriptors are retrieved through a class-specific Get String Command. See Section 5.3.5.1, “Class-specific String Descriptor ” for details. All class-specific strings in the Audio Function shall use this new methodology. Strings that are part of the standard Descriptor set shall use the standard string methodology.

Class-specific strings may be dynamic in nature, i.e., change during normal operation and inform the Host of such a change by generating an interrupt with source type set to STRING.

The **wLength** field contains the length of the class-specific String Descriptor. Class-specific strings may be up to 65,525 bytes in length.

The **wDescriptorType** field shall be set to EXT_STRING.

The **wDescriptorSubtype** field indicates the Descriptor subtype for the String Descriptor. (Currently, only the value STRING is defined.)

The **wDescriptorID** field contains a unique identifier for the class-specific String Descriptor in the range [256..65,535].

The **iLangID** field contains a zero-based index into the LANGID code array as returned by the Device. A Device can at most support 126 different languages since the LANGID code array is restricted to 254 bytes and each LANGID code takes up 2 bytes. The range of the **iLangID** is therefore from 0 to 125 maximum.

The **String** field contains the actual Unicode encoded string as outlined in the *USB Core Specifications*.

Table 4-54: Class-specific String Descriptor

Offset	Field	Size	Value	Description
0	wLength	2	Number	Size of the String Descriptor in bytes: 12+N.
2	wDescriptorType	2	Number	EXT_STRING. Type of this Descriptor.
4	wDescriptorSubtype	2	Constant	STRING Descriptor Subtype.
6	wDescriptorID	2	Number	Unique ID for this class-specific String Descriptor.
8	wStrDescriptorID	2	Number	Unique ID for a String Descriptor.
10	iLangID	1	Number	Zero-based index into the LANGID code array as returned by the Device.
11	bReserved	1	0x00	Shall be set to zero.
12	String	N	Number	Unicode UTF16LE encoded string. Follows the definitions outlined in the *USB Core Specifications*.

5 COMMANDS & REQUESTS

The following sections provides details about the Commands and Requests that are used to communicate and control the Audio Function.

5.1 STANDARD REQUESTS

The Audio Device Class supports the standard Requests described in Section 9, “USB Device Framework,” of the USB core specification. The Audio Device Class places no specific requirements on the values for the standard Requests.

5.2 CLASS-SPECIFIC AUDIO FUNCTION MANAGEMENT COMMANDS

Class-specific Audio Function Management Commands are used to manipulate global Audio Function Capabilities. These Commands are special in that they manipulate a global Audio Function Capability rather than a specific AudioControl within the Audio Function. These Capabilities are addressed through the Audio Function’s single AudioControl Interface. They do not use the normal Push/Pull Command paradigm (see further). Rather the underlying Set/Get Request paradigm is used to interact with these Capabilities as follows.

Table 5-1: Set/Get Function Request Layout

bmRequest Type	bRequest	wValue	wIndex	wLength	Data
00100001B	COMMIT SWITCH_FUNCTION	Zero	Zero and Interface	Length of Parameter Block	Parameter Block
10100001B	SWITCH_FUNCTION				

Bit D7 of the **bmRequestType** field shall be set to 0b0 for the Set Request and to 0b1 for the Get Request. It is a class-specific Request (D6..5 = 0b01), directed at an AudioControl Interface of the Audio Function (D4..0 = 0b00001).

The **bRequest** field shall contain the COMMIT or SWITCH_FUNCTION constant for the Set Request and the SWITCH_FUNCTION constant for the Get request. If the field contains a value other than the allowed values for this field, the Request shall return a Request Error.

The **wValue** field shall be set to zero. If the field contains a value other than zero, the Request shall return a Request Error.

The value in the low byte of the **wIndex** field shall be appropriate to the recipient. Only appropriate AudioControl Interface numbers may be used. If the Command specifies an unknown AudioControl Interface number, the Request shall return a Request Error.

The high byte of the **wIndex** field shall be set to zero. If it contains a value other than zero, the Request shall return a Request Error.

At this time, there are two global Capabilities defined as detailed below.

5.2.1 COMMIT

The Commit Capability is used to simultaneously update the CUR Attributes of all or a select group of AudioControls within the Audio Function with their corresponding preloaded NEXT Attribute values in a synchronized fashion. Only the Set Request is supported for this Capability. The Parameter Block shall contain

either zero or the **wDescriptorID** field value of a CommitGroup Descriptor. When zero is specified, all Armed AudioControl CUR attributes within the entire Audio Function are updated. When a CommitGroup Descriptor ID is specified, only the CUR Attributes of the Armed AudioControls that are Members of the indicated CommitGroup are updated. If the Parameter Block contains a value other than zero or a valid CommitGroup Descriptor ID, the Request shall return a Request Error.

If for some reason, one or more NEXT Attribute values have become invalid between the time the NEXT Attribute(s) were preloaded (and checked for their validity at that time), then the Commit Command shall return a Request Error and no updates to the CUR Attributes to any of the Armed AudioControls shall take place.

Note: This specification does not provide explicit means for the Host to find out which of the Armed AudioControls caused the Commit Command to return a Request Error. The Host should implement an appropriate strategy to recover gracefully from this situation.

The **wLength** field of the Set Request shall be set to two. If it contains a value other than two, the Request shall return a Request Error.

The Commiting by CommitGroup functionality is optional. Whether the Audio Function supports this option is indicated in the **dOptControls** field of the AC Self Descriptor (see Section 4.5.3.1, “AC Self Descriptor”).

Support for the Commit Capability is Conditionally Required. If none of the Audio Function’s AudioControls support the NEXT Attribute, then the Commit Capability shall not be supported. However, if one or more AudioControls do support the NEXT Attribute, then the Commit Capability shall be supported.

5.2.2 SWITCH FUNCTION

The Switch Function Capability is used to switch the entire Audio Function from its Base Revision Level operational mode into a Higher Revision Level operational mode. The Capability is a Read-Only/Write-Once (ROW1).

When read, the current Audio Function Revision Level, as indicated by the **bFunctionProtocol** in the Interface Association Descriptor, is returned as a single byte in the Parameter Block of the Get Request.

When written, the Parameter Block shall contain the single byte value as advertised in the **bFunctionProtocol** field of the Interface Association Descriptor of one of the HRL Interface Descriptor Sets, which is described in the BOS HRL Function Capability Descriptor in Section 8.2, “Discovery of Higher Version Level Support for an Audio Function8.2.” The Device shall not complete the Status Stage of the Set Request until the Audio Function has fully transitioned from BRL operation to HRL operation. Subsequent write operations to this AudioControl shall result in a Request Error. The only way for the Host to choose a different HRL operation mode (if more than one is supported by the Audio Function) is to reset the entire Device and restart operation in BRL mode.

The **wLength** field of the Set or Get Request shall be set to one. If it contains a value other than one, the Request shall return a Request Error.

Support for the Switch Function Control is Conditionally Required. If the Audio Function does not support an HRL operation mode, then this Request shall not be supported. However, If the Audio Function does support at least one HRL operation mode, then the Switch Function Control shall be supported.

5.3 CLASS-SPECIFIC AUDIOCONTROL COMMANDS

Class-specific AudioControl Commands are used to manipulate audio-related AudioControls. These AudioControls fall into two main groups: those that manipulate the Audio Function’s AudioControls, such as volume, tone, selector position, etc. and those that describe or influence data transfer over an AudioStreaming interface, such as the valid Alternate Settings Control.

- Control of an Audio Function is performed through the manipulation of the Attributes of individual AudioControls that are embedded in the Entities of the Audio Function. (The AudioControl Interface itself is considered to be an Entity in itself and uses Entity ID zero for access.)
  The class-specific AudioControl Interface Descriptor contains a collection of Entity Descriptors, each indicating which AudioControls are present in the Entity. Commands are always directed to the single AudioControl Interface of the Audio Function. The Command contains enough information (Entity ID, Control Selector, Control Attribute, etc.) for the Audio Function to decide to where a specific Command is to be routed.
- Control of the class-specific behavior of an AudioStreaming Interface is performed through manipulation of Interface AudioControls. Commands are directed to the AudioStreaming Interface where the AudioControl resides.

The Audio Device Class supports two additional class-specific Commands that do not manipulate AudioControls inside the Audio Function:

- String Commands provide a class-specific method to retrieve String Descriptors from the Audio Function. The String Command is introduced to overcome the USB core specification limitation that only provides for 255 Device-wide String Descriptors.
- Descriptor Commands provide a class-specific method to retrieve Descriptors from the Audio Function outside the standard Descriptor retrieval during enumeration. The Descriptor Command is introduced to overcome the USB core specification limitation that only provides for Descriptors that are a maximum of 255 bytes long. It also enables dynamically changing Descriptors (after enumeration).

In general, all AudioControls and their associated Commands are optional, unless explicitly stated otherwise in the AudioControl description (see Appendix A.25, “AudioControl Capabilities Overview” for an overview).

5.3.1 CLASS-SPECIFIC AUDIOCONTROL COMMAND STRUCTURE AND LAYOUT

To overcome the limitations imposed by the standard USB Requests, this specification introduces a new class-specific approach to communicate with the internals of the Audio Function. Standard Requests limit the amount of information that can be sent to the Device during the Setup transaction to four bytes (**wValue** and **wIndex** field of the Setup packet).

The new Push/Pull Command structure allows for much more information to be exchanged between the Host and the Device by moving the command related parameters from the Setup transaction to the Data transaction of the transfer.

The Push Command is used to send information to the Device and the Pull Command is used to retrieve information from the Device.

The Push Command consists of a single Set Request whereas the Pull Command is an atomic sequence of a Set Request, followed by a Get request. All three Requests closely follow the standard USB Request layout as defined in the *USB Core Specification*. The following table details their layout.

Table 5-2: Push and Pull Command Request Layout

bmRequest Type	bRequest	wValue	wIndex	wLength	Data
00100001B	PUSH PULL	Zero	Zero and Interface	Length of Parameter Block	Parameter Block
10100001B	PULL				

Bit D7 of the **bmRequestType** field shall be set to 0b0 for the Set Request and to 0b1 for the Get Request. It is a class-specific Request (D6..5 = 0b01), directed at an interface (AudioControl or AudioStreaming) of the Audio Function (D4..0 = 0b00001).

The **bRequest** field shall contain the PUSH constant for the Push Set Request. It shall contain the PULL constant for the Pull Set and Get Request. If the field contains a value other than these values respectively, the Request shall return a Request Error.

The **wValue** field shall be set to zero. If the field contains a value other than zero, the Request shall return a Request Error.

The value in the low byte of the **wIndex** field shall be appropriate to the recipient. Only appropriate Interface numbers may be used. If the Command specifies an unknown Interface number, the Request shall return a Request Error.

The high byte of the **wIndex** field shall be set to zero. If it contains a value other than zero, the Request shall return a Request Error.

If an Audio Function does not support a certain Command, it shall indicate this by returning a Request Error when that Command Set Request phase (see further) is issued to the Function. If a certain Push Command is supported, the associated Pull Command shall also be supported. Pull Commands may be supported without the associated Push Command being supported. If interrupts are supported, then all necessary Pull Commands shall be implemented that are required to retrieve the appropriate information from the Audio Function in response to these interrupts.

The following sections provide more details about the Push And Pull Commands and their associated Address parameters and Data.

5.3.1.1 PUSH COMMAND

The Push Command is used to send information to the Device. It uses the Parameter Block of a single Set Request to convey all the information needed by the Device to know how to execute the Push Command. This includes all necessary addressing information (AddressPart) as well as the new value(s) (DataPart) for the addressed item, if applicable. The Device shall not ACK the Set Request until it has changed all the internal state required by the Push Command. A Device indicates that it cannot fulfill the Push Command by returning a Request Error.

The length of the Parameter Block is indicated in the **wLength** field of the Set Request. The layout of the Parameter Block is described in the following section. If the parameter values are not supported, the Set Request shall return a Request Error.

5.3.1.1.1 PUSH COMMAND PARAMETER BLOCK

The Push Command Parameter Block is divided into two separate Parts. The AddressPart of the Parameter Block is always 12 bytes long and contains all addressing information for the Device to know which addressable item within the Audio Function is targeted. The DataPart of the Parameter Block contains the data value(s) for the targeted item. The layout of the DataPart is qualified by the parameters in the AddressPart. Refer to subsequent paragraphs for a detailed description of the DataPart of the Parameter Block for all possible addressable items.

Table 5-3: Push Command Parameter Block Layout

Offset	Field	Size	Value	Description
AddressPart				

Offset	Field	Size	Value	Description
0	wParam1	2	Number	Command Parameter 1.
2	wParam2	2	Number	Command Parameter 2.
4	wParam3	2	Number	Command Parameter 3.
6	wParam4	2	Number	Command Parameter 4.
8	wParam5	2	Number	Command Parameter 5.
10	wParam6	2	Number	Command Parameter 6.
DataPart				
12	Data	n	Number	The data value(s).

5.3.1.2 PULL COMMAND

The Pull Command is used to retrieve information from the Device. The Pull Command always consists of a Set Request followed by its associated Get Request without an intervening Set Request to the same item. The Parameter Block of the Set Request contains all addressing information (AddressPart) for the Device to know for which item data is being requested by the Host. The Parameter Block of the associated Get Request contains the actual requested Device data (DataPart). If the Device receives a Pull Command Get Request without having received a Pull Command Set Request first, it shall indicate this by returning a Request Error on the Pull Command Get Request.

5.3.1.2.1 PULL COMMAND SET REQUEST PARAMETER BLOCK

The Pull Command Set Request Parameter Block consists of the AddressPart only. It contains all addressing information for the Device to know which addressable item within the Audio Function is targeted. The DataPart is retrieved from the Device in the parameter Block of the associated atomic Pull Command Get Request.

The length of the Parameter Block is indicated in the **wLength** field of the Pull Command Set Request. This field shall always be set to a value of 12. If the parameter values are not supported, the Set Request shall return a Request Error.

The layout of the Pull Command Set Request Parameter Block is as follows.

Table 5-4: Pull Command Set Request Parameter Block Layout

Offset	Field	Size	Value	Description
0	wParam1	2	Number	Command Parameter 1.
2	wParam2	2	Number	Command Parameter 2.
4	wParam3	2	Number	Command Parameter 3.
6	wParam4	2	Number	Command Parameter 4.
8	wParam5	2	Number	Command Parameter 5.
10	wParam6	2	Number	Command Parameter 6.

5.3.1.2.2 PULL COMMAND GET REQUEST PARAMETER BLOCK

The data value(s) for the Pull Command are returned in the Parameter Block of the Pull Command Get Request. It consists of the DataPart only and contains the data value(s) for the targeted item.

The length of the Parameter Block to return is indicated in the **wLength** field of the Get Request. If the Parameter Block is longer than what is indicated in the **wLength** field, only the initial bytes of the Parameter Block are returned. If the Parameter Block is shorter than what is indicated in the **wLength** field, the Device indicates the end of the control transfer by sending a short packet when further data is requested. The layout of the Get Request Parameter Block is qualified by the parameters in the Address Parameter Block of the associated Pull Command Set Request. Refer to subsequent paragraphs for a detailed description of the DataPart for all possible addressable items.

The layout of the Pull Command Get Request Parameter Block is as follows.

Table 5-5: Pull Command Get Request Parameter Block Layout

Offset	Field	Size	Value	Description
DataPart				
0	Data	n	Number	Data value(s) returned by the Device.

The remainder of this section describes the class-specific Commands and their characteristics used to manipulate the incorporated AudioControls, class-specific Strings and Descriptors.

5.3.2 AUDIOCONTROL COMMANDS CHARACTERISTICS

When responding to an AudioControl Push Command, an Audio Function shall not complete the Status Stage of the Control Transfer until the Command has been successfully interpreted and the Audio Function has successfully completed the associated action. The Host always has the option to abort or abandon a pending Control Transfer if it deems to take too long to complete.

The following sections describe the possible Commands that can be used to manipulate the AudioControls an Audio Function exposes through its Interfaces, Endpoints, and Entities. The same layout of the Parameter Blocks is used for both the Push and Pull Commands.

5.3.2.1 AUDIOCONTROL INTERRUPT CAPABILITIES

An AudioControl shall generate an interrupt when any of its Attributes change other than through Host manipulation.

5.3.2.2 AUDIOCONTROL ATTRIBUTES

Each AudioControl within an Entity may have one or more Attributes associated with it. Currently defined Attributes for an AudioControl are its:

- Current Attribute (CUR)
- Next Attribute (NEXT)
- Capabilities Attribute (CAP)
- Range Attribute (RANGE)

Attributes are manipulated by issuing the appropriate Push and Pull Commands to the targeted AudioControl as detailed below.

5.3.2.2.1 CURRENT AND NEXT ATTRIBUTES

The Current (CUR) Attribute is used to manipulate the current setting of an AudioControl. Manipulating this Attribute has immediate effect on the actual setting of the AudioControl.

In some cases, it is desirable to apply settings to multiple AudioControls simultaneously. For example, changing the settings for a Parametric Equalizer Section (Center Frequency, Q Factor, and Gain) sequentially rather than simultaneously may introduce totally undesired side effects and artifacts. To cover these cases, this specification supports the NEXT Attribute and the Commit Command. Manipulating the NEXT Attribute Arms the AudioControl with a next value without changing its CUR Attribute; this AudioControl is said to be Armed. The Commit Command will only complete successfully after the NEXT Attributes of all the Armed AudioControls have been applied to their respective CUR Attributes. This applies to either all AudioControls that have been updated since the last Commit Command, or only to those AudioControls that are a Member of the CommitGroup that is indicated in the Commit Command. This effectively changes all the affected AudioControls in the Audio Function simultaneously (within the limits of the underlying firmware and hardware) and the NEXT and CUR Attributes of the affected AudioControls will now have the same value. After the Commit Command completes, either successfully or resulting in a request error, all affected Armed AudioControls return to the Unarmed state and subsequent Commit Commands have no impact on these AudioControls unitl their NEXT Attribute is updated again.

Note that manipulating the CUR Attribute of an AudioControl that has a preloaded NEXT Attribute, does NOT change the value of its NEXT Attribute. Whenever the value of the CUR Attribute is changed, the NEXT Attribute retains its preloaded value. This means that a subsequent Commit Command will overwrite the value of the CUR attribute that was issued between the write action to the NEXT Attribute and the exercution of the Commit Command.

AudioControls that are intended to be manipulated simultaneously can declare their CUR Attribute as Read (r) and solely rely on the update mechanism described above (staging all the NEXT Attributes involved, followed by a Commit Command).

Whenever the USB Device containing the Audio Function receives a Set Configuration Request, all Armed AudioControls shall no longer be Armed.

5.3.2.2.2 CAPABILITIES ATTRIBUTE

The mandatory Capabilities (CAP) Attribute is used to retrieve all necessary information from the AudioControl for the Host software to determine how to successfully interact with the AudioControl. The data structure returned via a Get(CAP) Command is as follows.

Bit D0 of the **bmControlCaps** field indicates the implemented support level for the CUR Attribute.

- For AudioControls that are allowed by specification to have the CUR Attribute implemented as Read-Only (RO) or Read-optional-Write (RoW), bit D0 set to 0b0 indicates that the Audio Function has implemented the CUR Attribute as Read (r). If D0 is set to 0b1, then the Audio Function has implemented the CUR Attribute as Read-Write (rw).
- For AudioControls that are allowed by specification to have the CUR Attribute implemented as Write-Only (WO) bit D0 shall be set to 0b0.
- For AudioControls that are required by specification to have the CUR Attribute implemented as mandatory-Read-Write (mRW), bit D0 shall be set to 0b1.

Bit D1 of the **bmControlCaps** field indicates whether the NEXT Attribute is implemented (D1 = 0b1) or not (D1 = 0b0). If implemented, it shall always be implemented as Read-Write (rw).

Bit D2 of the **bmControlCaps** field indicates whether the RANGE Attribute uses the Triplet format (D2 = 0b0) or the ValueList format (D2 = 0b1). Bit D2 shall be set to 0b0 when the RANGE Attribute is Prohibited for the AudioControl.

Bit D3 of the **bmControlCaps** field indicates whether the AudioControl is calibrated for a specific use (D3 = 0b1) or not (D3 = 0b0). When set, bit D3 indicates that there is a preferred value of the AudioControl that is calibrated such that the resulting effect on the audio stream is deemed the preferred level of impact for that specific use by the manufacturer.

Currently the use of Bit D3 is only supported for Gain Controls that are in a microphone signal path. See Section 5.3.3.5.3, “Gain Control” for details. The use of this bit for other AudioControls is reserved for future use.

Bit D4 of the **bmControlCaps** field indicates whether the AudioControl is intended by the manufacturer to be hidden from the user (D4 = 0b1) or not (D4 = 0b0).

Note: The CUR and CAP Attributes shall always be implemented and therefore, there is no need to express this in the **bmControlCaps** field.

Table 5-6: Capabilities Attribute DataPart

wLength		1		
Offset	**Field**	**Size**	**Value**	**Description**
0	bmControlCaps	1	Bitmap	D0: CUR Attribute implementation support level. 0 = Read (r) for RoW Controls or Write (w) for WO Controls. 1 = Read-Write (rw). D1: NEXT Attribute Implementation. 0 = Not Implemented. 1 = Implemented. D2: RANGE Attribute Format. 0 = Array of Triplets. 1 = ValueList. D3: Calibrated AudioControl. 0 = No. 1 = Yes. D4: User-Hidden. 0 = No. 1 = Yes. D7..5: Reserved.

5.3.2.2.3 RANGE ATTRIBUTE

The RANGE Attribute provides information about the limitations the AudioControl imposes on the allowed settings of the CUR and NEXT Attributes. The RANGE Attribute either consists of an array of sub-ranges or an enumeration of possible values. The format a particular AudioControl uses for its RANGE Attribute is indicated by the RANGE Attribute Format bit in the AudioControl’s Capabilities Attribute.

If the RANGE Attribute Format bit indicates “Array of Triplets”, then sub-ranges are described via the Minimum (MIN), Maximum (MAX), and Resolution (RES) fields. They are always grouped in triplets of the form [MIN, MAX,

RES]. The RANGE Attribute supports an array of these triplets so that discontinuous multiple subranges can be accurately reported. The first element in the Parameter Block contains the number of subranges the AudioControl supports. Subsequent triplet elements in the Parameter Block correspond to each of the subranges. The subranges shall be ordered in ascending order (from lower values to higher values). Individual subranges shall not overlap (i.e., the MAX value of the previous subrange shall be less than the MIN value of the next subrange). If a subrange consists of only a single value, the corresponding triplet shall contain that value for both its MIN and MAX sub-Attribute and the RES sub-Attribute shall be set to zero.

If the RANGE Attribute Format bit indicates “ValueList”, then the RANGE Attribute exposes an enumeration of possible values for the CUR and NEXT Attributes. The first value in the list indicates the number of elements in the enumeration.

In all cases, the values returned by the RANGE Attribute shall use the same format as the CUR and NEXT Attributes as defined by this specification.

As an example, consider a (hypothetical) Control that takes the following values for its CUR Attribute:

- -∞ dB
- -70 dB to -40 dB in steps of 3 dB
- -38 dB to -20 dB in steps of 2 dB
- -19 dB to 0 dB in steps of 1 dB

One possible layout of the RANGE Attribute is then:

RANGE Attribute Format = 0
wNumSubRanges = 3
RANGE(1) = [-70, -40, 3]
RANGE(2) = [-38, -20, 2]
RANGE(3) = [-19, 0, 1]

Another way of representing the same Control is:

wNumSubRanges = 3
RANGE(1) = [-70, -43, 3]
RANGE(2) = [-40, -22, 2]
RANGE(3) = [-20, 0, 1]

It is left to the designer to choose a suitable representation.

As a second example, consider a Mixer Control that is implemented as Read (r) and that has a fixed value of -6 dB. The RANGE Attribute for that Mixer Control is then:

Range Attribute Format = 1
wNumValues = 1
Value(1) = -6

In the Sections that describe the AudioControls in detail, the support level for the AudioControl Attributes is explicitly called out.

5.3.2.3 AUDIOCONTROL READ/WRITE PRIVILEGES

Note: It is important to make a distinction between the Read/Write privilege allowed by the specification and the actual Read/Write privilege an Audio Function chooses to implement for an AudioControl’s

Attributes. The *specification* privileges are expressed as Read-Only (RO) or Read-optional-Write (RoW) for AudioControl Attributes that may be *implemented* as either Read (r) or Read-Write (rw) respectively, and mandatory-Read-Write (mRW) for AudioControl Attributes that shall be *implemented* as Read-Write (rw). The *specification* privileges are further expressed as Write-Only (WO) for AudioControl Attributes that shall be *implemented* as Write (w).

The actual Read/Write privilege an Audio Function chooses to implement for an AudioControl's Attributes (indicated by the abbreviations (r) for Read, (w) for Write, and (rw) for Read-Write) can be retrieved via the mandatory CAP Attribute for each AudioControl. For details, see Section 5.3.2.2.2, "Capabilities Attribute."

This specification defines the Read/Write privileges of an AudioControl based on the Read/Write privilege of its CUR Attribute as follows:

- Some AudioControls are defined as Read-Only (RO). For this type of AudioControls, the CUR Attribute shall be Read (r) and the NEXT Attribute shall not be supported. It is recommended that the RANGE Attribute be supported, if only to assist the Host with meaningful information for UI display purposes.
- Most AudioControls are defined as Read-optional-Write (RoW). This means that a particular implementation can decide whether to implement the AudioControl's CUR Attribute either as Read-Write (rw) or as Read (r). Most AudioControls that implement their CUR Attribute as Read (r) usually do not support the NEXT Attribute. However, some AudioControls may implement their CUR Attribute as Read (r) and provide a NEXT Attribute (always implemented as Read-Write (rw)) and rely on the presence and use of the Commit Control to change the current value. In other words, these AudioControls cannot have the value of their CUR Attribute changed by the driver directly but only through an update via the NEXT Attribute followed by a Commit. For AudioControls that are implemented as Read-Write (rw), the NEXT Attribute may be supported and shall always be implemented as Read-Write (rw).
- The Commit Control is the only AudioControl defined as Write-Only (WO) by the current specification. The Commit Control's CUR Attribute shall be implemented as Write (w) and the NEXT Attribute shall not be supported. The RANGE Attribute shall not be supported for this AudioControl.
- All AudioControls shall support the Capabilities Attribute and it shall be implemented as Read (r).
- All AudioControls shall have the RANGE Attribute, either explicitly implemented by advertising MIN, MAX, and RES or VALUELIST values. However, some AudioControls have a range that is fixed by this specification or inherent by their type. These AudioControls shall not advertise a RANGE Attribute. All Boolean type AudioControls fall into this category, for example. Explicitly advertised RANGE attributes shall always be implemented as Read (r). Note, however, that an implementation is allowed to update the value(s) of the RANGE attribute as a result of an external event.

In summary, the specification may allow the Read/Write privilege of an Attribute to be Read-Only (RO), Read-optional-Write (RoW), mandatory-Read-Write (mRW), or Write-Only (WO). An implementation shall always express the Read/Write privilege of an Attribute as either Read (r), Read-Write (rw), or Write (w).

5.3.2.4 AUDIOCONTROL COMMAND ADDRESSPART LAYOUT

The AudioControl Push and Pull Commands are used to manipulate an Attribute of an AudioControl inside the Audio Function. The following table details the AddressPart layout.

Table 5-7: AudioControl Command AddressPart Layout

Offset	Field	Size	Value	Description
0	wEntityID	2	Number	Entity ID of the addressed Item.
2	wCS	2	Constant	Control Selector for the targeted Item.
4	wAttribute	2	Constant	Attribute of the targeted Item.
6	wOCN	2	Number	Output Channel Number on which the targeted Item resides.
8	wICN	2	Number	Input Channel Number on which the targeted Item resides.
10	wIPN	2	Number	Input Pin Number on which the targeted Item resides.

The **wEntityID** field shall contain the unique ID of the Entity (Clock Entity ID, Unit ID, Terminal ID, Power Domain ID, or Connector ID) within which the targeted AudioControl resides. When addressing AudioControls residing within an Interface, the **wEntityID** field shall be set to zero. The values in the **wEntityID** field shall be appropriate to the recipient. Only existing Entities in the Audio Function or in the AudioStreaming Interfaces may be used. If the Command specifies an unknown or non-Entity ID, the Command Set Request shall return a Request Error.

The **wCS** field specifies the Control Selector (CS). The Control Selector indicates which type of AudioControl this Command is manipulating. If the Command specifies an unknown or unsupported CS for the targeted Entity, the Command Set Request shall return a Request Error.

The **wAttribute** field contains a constant, identifying which Attribute of the targeted AudioControl is to be manipulated. Possible Attributes for an AudioControl are its:

- Current Attribute (CUR)
- Next Attribute (NEXT)
- Capabilities Attribute (CAP)
- Range Attribute (RANGE)

If the targeted AudioControl does not support modification of a certain Attribute, the Command Set Request shall return a Request Error when an attempt is made to modify that Attribute. In most cases, only the CUR Attribute will be supported for the Push Command.

This specification does not prevent a designer from having the Audio Function adjust a RANGE Attribute due to an external event and alert the Host of this change through an interrupt.

For the list of Request constants, refer to Appendix A.22, “Class-specific Attribute Codes.”

The **wOCN** field specifies the Output Channel Number (OCN). The OCN is equal to the number of the Output Pin Channel on which the AudioControl resides. If an AudioControl is output channel independent, then the OCN shall be set to zero (OCN = 0). If the Command specifies an unknown or unsupported OCN for the targeted Entity, the Command Set Request shall return a Request Error.

The **wICN** field specifies the Input Channel Number (ICN). The ICN is equal to the number of the Input Pin Channel on which the AudioControl resides. If an AudioControl is input channel independent, then the ICN shall be set to zero (ICN = 0). If the Command specifies an unknown or unsupported ICN for the targeted Entity, the Command Set Request shall return a Request Error.

The **wIPN** field specifies the Input Pin Number (IPN). The IPN is equal to the Input Pin Number on which the AudioControl resides. If an AudioControl is Input Pin independent, then the IPN shall be set to zero (IPN = 0). If the Command specifies an unknown or unsupported IPN for the targeted Entity, the Command Set Request shall return a Request Error.

The OCN:ICN:IPN triplet shall always reflect the correct values, i.e., for a Unit that only has a single Input Pin, the IPN shall be set to one, and the ICN shall be set equal to the OCN since the input channel and the output channel are exactly the same.

5.3.2.5 AUDIOCONTROL COMMAND DATAPART LAYOUT

With a few exceptions, almost all AudioControl Commands manipulate a single AudioControl Attribute parameter during a Push or Pull Command. For those Commands, the possible DataPart layouts for the CUR, NEXT, and RANGE Attributes can be divided into three layout categories, depending on the byte size of the AudioControl's CUR Attributes. See further for the Layout 1 (Section 5.3.2.6.1, "Layout 1 DataPart"), Layout 2 (Section 5.3.2.6.2, "Layout 2 DataPart"), and Layout 3 (Section 5.3.2.6.3, "Layout 3 DataPart") definitions. The layout for the CAP Attribute is defined in Section 5.3.2.2.2, "Capabilities Attribute". The following paragraphs specify the layout of the CUR, NEXT, and RANGE DataParts for the three categories.

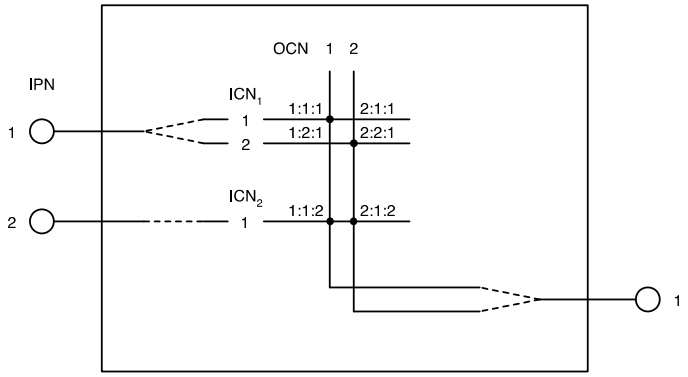
For those Commands that use a custom DataPart layout, the actual layout is explicitly defined in the relevant sections.

5.3.2.6 USE OF WILDCARDS IN THE OCN:ICN:IPN TRIPLET

This specification allows for a wildcard value to be used in any or all of the values in the OCN:ICN:IPN triplet. The wildcard value shall be 0xFFFF. When a value of the OCN:ICN:IPN triplet is set to the wildcard value, this means that the Attribute of all AudioControls in the range of that wildcard triplet value are manipulated at the same time. The DataPart shall contain an Attribute value for all the AudioControls in the range of that wildcard triplet. Note that only Attribute values for AudioControls that are actually implemented in the Entity, as indicated by the **d(a)BmOptControls** field(s) in the Entity Descriptor, shall be present in the DataPart.

The ordering of the values in the DataPart is from lowest channel number to highest channel number, OCN first, followed by ICN, followed by IPN. As a hypothetical example, consider an Entity that has 2 Input Pins, 2 input channels on Input Pin 1, 1 input channel on Input Pin 2, and 2 channels on its Output Pin. Assume further that AudioControls are present as indicated by the dots in the following figure.

Figure 5-1: OCN:ICN:IPN Example



For this Entity, the DataPart for the CUR Attribute of the AudioControl would look like this for the following uses of the wildcard:

OCN:ICN:IPN = 0xFFFF:0xFFFF:0xFFFF: $CUR_{1:1:1}$; $CUR_{2:2:1}$; $CUR_{1:1:2}$; $CUR_{2:1:2}$

OCN:ICN:IPN = 0xFFFF:0xFFFF:0x0001: $CUR_{1:1:1}$; $CUR_{2:2:1}$

OCN:ICN:IPN = 0xFFFF:0x0001:0xFFFF: $CUR_{1:1:1}$; $CUR_{1:1:2}$; $CUR_{2:1:2}$

OCN:ICN:IPN = 0x0001:0xFFFF:0xFFFF: $CUR_{1:1:1}$; $CUR_{1:1:2}$

OCN:ICN:IPN = 0xFFFF:0x0002:0x0001: $CUR_{2:2:1}$

Other Attributes can be retrieved in the same fashion. It is the responsibility of the Host to check whether the expected volume of information will fit within the limitations of the DataPart maximum size (65535 – 12 = 65523 bytes). Note that for the RANGE Attribute, the actual size of the information returned for each AudioControl may not be known beforehand. Therefore, the use of wildcards while manipulating the RANGE Attribute needs to be approached with caution.

5.3.2.6.1 LAYOUT 1 DATAPART

The DataPart for a 1-byte sized CUR or NEXT Attribute of an AudioControl is as follows:

Table 5-8: 1-byte AudioControl CUR or NEXT DataPart

wLength		1		
Offset	**Field**	**Size**	**Value**	**Description**
0	bCUR bNEXT	1	Number	The setting for the CUR or NEXT Attribute of the targeted AudioControl.

The associated DataPart for the RANGE Attribute of that AudioControl when expressed as an Array of Triplets is as follows:

Table 5-9: 1-byte AudioControl RANGE DataPart (Array of Triplets)

wLength		2+3*n		
Offset	**Field**	**Size**	**Value**	**Description**
0	wNumSubRanges	2	Number	The number of subranges of the targeted AudioControl: n.
2	bMIN(1)	1	Number	The setting for the MIN Attribute of the first subrange of the targeted AudioControl.
3	bMAX(1)	1	Number	The setting for the MAX Attribute of the first subrange of the targeted AudioControl.
4	bRES(1)	1	Number	The setting for the RES Attribute of the first subrange of the targeted AudioControl.
…	…	…	…	…
2+3*(n-1)	bMIN(n)	1	Number	The setting for the MIN Attribute of the last subrange of the targeted AudioControl.
3+3*(n-1)	bMAX(n)	1	Number	The setting for the MAX Attribute of the last subrange of the targeted AudioControl.

wLength		2+3*n		
4+3*(n-1)	bRES(n)	1	Number	The setting for the RES Attribute of the last subrange of the targeted AudioControl.

The associated DataPart for the RANGE Attribute of that AudioControl when expressed as a ValueList is as follows:

Table 5-10: 1-byte AudioControl RANGE DataPart (ValueList)

wLength		2+n		
Offset	**Field**	**Size**	**Value**	**Description**
0	wNumValues	2	Number	The number of values in the ValueList: n.
2	bValue(1)	1	Number	The first value in the ValueList for the targeted AudioControl.
…	…	…	…	…
2+(n-1)	bValue(n)	1	Number	The last value in the ValueList for the targeted AudioControl.

5.3.2.6.2 LAYOUT 2 DATAPART

The DataPart for a 2-byte sized CUR or NEXT Attribute of an AudioControl is as follows:

Table 5-11: 2-byte AudioControl CUR or NEXT DataPart

wLength		2		
Offset	**Field**	**Size**	**Value**	**Description**
0	wCUR wNEXT	2	Number	The setting for the CUR or NEXT Attribute of the targeted AudioControl.

The associated DataPart for the RANGE Attribute of that AudioControl when expressed as an Array of Triplets is as follows:

Table 5-12: 2-byte AudioControl RANGE DataPart (Array of Triplets)

wLength		2+6*n		
Offset	**Field**	**Size**	**Value**	**Description**
0	wNumSubRanges	2	Number	The number of subranges of the targeted AudioControl: n.
2	wMIN(1)	2	Number	The setting for the MIN Attribute of the first subrange of the targeted AudioControl.
4	wMAX(1)	2	Number	The setting for the MAX Attribute of the first subrange of the targeted AudioControl.
6	wRES(1)	2	Number	The setting for the RES Attribute of the first subrange of the targeted AudioControl.
…	…	…	…	…
2+6*(n-1)	wMIN(n)	2	Number	The setting for the MIN Attribute of the last subrange of the targeted AudioControl.
4+6*(n-1)	wMAX(n)	2	Number	The setting for the MAX Attribute of the last subrange of the targeted AudioControl.

wLength		2+6*n		
6+6*(n-1)	wRES(n)	2	Number	The setting for the RES Attribute of the last subrange of the targeted AudioControl.

The associated DataPart for the RANGE Attribute of that AudioControl when expressed as a ValueList is as follows:

Table 5-13: 2-byte AudioControl RANGE DataPart (ValueList)

wLength		2+2*n		
Offset	**Field**	**Size**	**Value**	**Description**
0	wNumValues	2	Number	The number of values in the ValueList: n.
2	wValue(1)	1	Number	The first value in the ValueList for the targeted AudioControl.
…	…	…	…	…
2+2*(n-1)	wValue(n)	1	Number	The last value in the ValueList for the targeted AudioControl..

5.3.2.6.3 LAYOUT 3 DATAPART

The DataPart for a 4-byte sized CUR or NEXT Attribute of an AudioControl is as follows:

Table 5-14: 4-byte AudioControl CUR or NEXT DataPart

wLength		4		
Offset	**Field**	**Size**	**Value**	**Description**
0	dCUR dNEXT	4	Number	The setting for the CUR or NEXT Attribute of the targeted AudioControl.

The associated DataPart for the RANGE Attribute of that AudioControl when expressed as an Array of Triplets is as follows:

Table 5-15: 4-byte AudioControl RANGE DataPart (Array of Triplets)

wLength		2+12*n		
Offset	**Field**	**Size**	**Value**	**Description**
0	wNumSubRanges	2	Number	The number of subranges of the targeted AudioControl: n
2	dMIN(1)	4	Number	The setting for the MIN Attribute of the first subrange of the targeted AudioControl.
6	dMAX(1)	4	Number	The setting for the MAX Attribute of the first subrange of the targeted AudioControl.
10	dRES(1)	4	Number	The setting for the RES Attribute of the first subrange of the targeted AudioControl.
…	…	…	…	…
2+12*(n-1)	dMIN(n)	4	Number	The setting for the MIN Attribute of the last subrange of the targeted AudioControl.
6+12*(n-1)	dMAX(n)	4	Number	The setting for the MAX Attribute of the last subrange of the targeted AudioControl.

wLength		2+12*n		
10+12*(n-1)	dRES(n)	4	Number	The setting for the RES Attribute of the last subrange of the targeted AudioControl.

The associated DataPart for the RANGE Attribute of that AudioControl when expressed as a ValueList is as follows:

Table 5-16: 4-byte AudioControl RANGE DataPart (ValueList)

wLength		2+4*n		
Offset	**Field**	**Size**	**Value**	**Description**
0	wNumValues	2	Number	The number of values in the ValueList: n.
2	dValue(1)	1	Number	The first value in the ValueList for the targeted AudioControl..
…	…	…	…	…
2+4*(n-1)	dValue(n)	1	Number	The last value in the ValueList for the targeted AudioControl

5.3.2.7 AUDIOCONTROL SECTIONS LAYOUT

The layout of the following sections that describe the possible AudioControls in more detail is as follows:

First, there is a paragraph briefly describing the functionality of the AudioControl. Most AudioControls are optional for implementations to support. However, some AudioControls shall be implemented by all class-compliant Audio Function implementations. Some AudioControls may be required under certain conditions. All AudioControls are clearly marked as Mandatory, Optional, or Conditionally Required in the header of the AudioControl Table, using the abbreviations MAN, OPT, or CR respectively (see below). If an AudioControl is marked as Conditionally Required, then the conditions are explained in the descriptive paragraph preceding the AudioControl Table.

Note: The mandatory or optional character of the AudioControl is qualified by the fact whether the Audio Function implements a certain Entity that contains the AudioControl. For example, the Power Domain Control is marked as Mandatory, but obviously it is only mandatory if the Audio Function implements a Power Domain.

Then a standardized AudioControl Table follows that summarizes several different aspects of the AudioControl and applicable values for various fields in the AudioControl Command that is used to manipulate the AudioControl.

The first rows of the table enumerate the possible Attributes (CUR, NEXT). For each Attribute, the required support level *for the Attribute* is indicated in the SL (Support Level) column. Mandatory (M) indicates that the AudioControl shall support this Attribute. Optional (O) indicates that the AudioControl may choose to support the Attribute, while Prohibited (P) indicates that the AudioControl is not allowed to support the Attribute. Implicit (I) indicates that the Attribute support is fixed by the specification or by the type of the AudioControl.

The RW column indicates the level of freedom (allowed by this specification) an implementation has regarding the Read-Write privilege of the Attribute: whether the Attribute can be implemented as Read-Only (RO), mandatory-Read-Write (mRW), Read with optional Write (RoW), or Write-Only (WO).

The RANGE rows together specify the range of values that are applicable for the CUR and NEXT Attribute. The VALUE LIST row specifies a list of values that is applicable for the CUR and NEXT Attribute.

The acronym N/A is used to indicate “Not Applicable”.

The CS row contains the Control Selector value that shall be used for that AudioControl.

The OCN:ICN:IPN row contains information about the values that can be used to target a particular AudioControl within the Entity.

Finally, if the AudioControl has a custom DataPart layout (other than the default DataPart layouts), that layout is explicitly listed in subsequent rows, as applicable.

The following syntax is used to indicate values and value ranges:

[value] ::= single value

[value1 to value2] ::= a single value in the range between and including value1 and value2

[value1 to value2]+ ::= list of values, containing one or more elements in the range between and including value1 and value2

5.3.2.7.1 EXAMPLE 1: AUDIOCONTROL TABLE WITH DEFAULT DATAPART LAYOUT

Table 5-17: AudioControl Table with Default DataPart Layout

Name	SL	RW	MAN/OPT/CR	
CUR	M	RO/RoW/mRW/WO		
NEXT	M/O/P	mRW		
RANGE	M/I	RO	MIN	Applicable range
			MAX	Applicable range
			RES	Applicable range
			VLIST	Applicable Value List
Field	**Value**			
CS	Control Selector Value			
OCN:ICN:IPN	As applicable			
Parameter Block	Layout 1, 2, or 3			

5.3.2.7.2 EXAMPLE 2: AUDIOCONTROL TABLE WITH CUSTOM DATAPART LAYOUT

Table 5-18: AudioControl Table with Custom DataPart

Name	SL	RW	MAN/OPT/CR			
CUR	M	RO/RoW/mRW/WO				
NEXT	M/O/P	mRW				
RANGE	M/I	RO	MIN	Applicable range		
			MAX	Applicable range		
			RES	Applicable range		
			VLIST	Applicable Value List		
Field	**Value**					
CS	Control Selector Value					
OCN:ICN:IPN	As applicable					
Parameter Block	Length	Length in bytes of the CUR or NEXT custom DataPart				
Offset	**Field**		**Size**	**Value**	**Description**	
0	Field Name		x	As Applicable	…	
…	…		…	…	…	

5.3.3 AUDIOCONTROL COMMANDS

The following sections describe the Commands an Audio Function may support for its AudioControl Interface. The same layout of the DataParts is used for both the Push and Pull Commands.

AudioControl Commands are directed to the Entity that contains the targeted AudioControl via the single AudioControl Interface of the Audio Function.

5.3.3.1 COMMON AUDIOCONTROLS

The following sections describe AudioControls that may appear in several Entity types. They are described here only once and a reference to these AudioControl descriptions is provided for all those Entities that may incorporate any of these AudioControls.

5.3.3.1.1 BYPASS CONTROL

The Bypass Control is implemented either on each Pin channel intersection individually or on the Unit as whole. In the former case, it is used to include the functionality offered by an Entity on a particular OCN:ICN:IPN Node or to bypass that functionality on that Node. In the latter case, it used to either include the functionality of an Entity in the signal path or bypass the Entity entirely (OCN=ICN=IPN = 0). The input Cluster on Input Pin one is then routed unaltered to the output of the Entity.

For most Entities, the Bypass Control is optional, for some it is required to be implemented.

Table 5-19: Bypass Control Characteristics

Bypass	**SL**	**RW**	OPT or MAN	
CUR	M	RoW	[0 (0x00) to 1 (0x01)]; Active or Bypassed respectively.	
NEXT	O	mRW		
RANGE	I	N/A	**MIN**	N/A
			MAX	N/A
			RES	N/A
			VLIST	N/A
Field	**Value**			
CS	XX_BYPASS_CONTROL (where XX shall be replaced by the appropriate two-letter abbreviation for the particular Entity)			
OCN:ICN:IPN	As applicable			
Parameter Block	Layout 1 (See Section 5.3.2.6.1, “Layout 1 DataPart”)			

5.3.3.1.2 CLUSTER CONTROL

The Cluster Control is used to select among different logical Cluster Configurations the Entity may support on its Output Pin. A value of zero indicates that the Cluster *Configuration* is inherited from the Entity’s Input Pin 1. For an Input Terminal (that has no Input Pin 1), a value of zero indicates that its associated AudioStreaming Interface is currently set to Alternate Setting zero and consequently, there is no Cluster Configuration available. This also means that the Active Cluster Control of the Input Terminal shall report that the Cluster is Inactive.

Whenever this AudioControl is present, the Cluster Active Control shall also be present. It is important to note that this does not mean that the Cluster *content* or the state of the Cluster Active Control is directly inherited from Input Pin 1. The range for this AudioControl is from 0 to the number of Clusters, supported on the Output Pin of the Entity, as reflected in the Entity’s **wNrClusters** Descriptor field.

Table 5-20: Cluster Control Characteristics

Mode Select	SL	RW	OPT	
CUR	M	RoW	[0 to **bNrClusters**]	
NEXT	O	mRW		
RANGE	I	N/A	MIN	N/A
			MAX	N/A
			RES	N/A
			VLIST	N/A
Field	**Value**			
CS	XX_CLUSTER_CONTROL (where XX shall be replaced by the appropriate two-letter abbreviation for the particular Entity)			
OCN:ICN:IPN	0:0:0			
Parameter Block	Layout 1 (See Section 5.3.2.6.1, “Layout 1 DataPart”)			

5.3.3.1.3 CLUSTER ACTIVE CONTROL

The Read-Only Cluster Active Control indicates whether the Cluster present on the Output Pin of the Entity is currently Active or Inactive. The Entity that contains this AudioControl shall decide by itself which value to report, based on a variety of criteria, including the state of the Clusters being received on its Input Pin(s).

Table 5-21: Cluster Active Control Characteristics

Cluster Active	SL	RW	MAN	
CUR	M	RO	[0 (0x00) to 1 (0x01)]; Active or Inactive respectively.	
NEXT	P	N/A		
RANGE	I	N/A	MIN	N/A
			MAX	N/A
			RES	N/A
			VLIST	N/A
Field	**Value**			
CS	XX_CLUSTER_ACTIVE_CONTROL (where XX shall be replaced by the appropriate two-letter abbreviation for the particular Entity)			
OCN:ICN:IPN	0:0:0			
Parameter Block	Layout 1 (See Section 5.3.2.6.1, “Layout 1 DataPart”)			

5.3.3.1.4 LATENCY CONTROL

An Audio Function shall either not support this AudioControl (D0 = 0b0 in the **bmOptControls** field of the class-specific AudioControl Interface Descriptor) or support this Read-Only AudioControl for *every* Terminal and Unit within the Audio Function (D0 = 0b1 in the **bmOptControls** field of the class-specific AudioControl Interface Descriptor). Terminal latencies shall include all latencies incurred by A/D or D/A converters, encoders, decoders, etc.

Note: The presence of the Latency Controls is advertised in the class-specific AudioControl Interface Descriptor and not repeated in every Terminal and Unit Descriptor. Its functionality is described here, although the AudioControl Interface by itself does not contain a Latency Control.

Table 5-22: Latency Control Characteristics

Latency	SL	RW	OPT	
CUR	M	RO	[0 ns (0x00000000) to 4,294,967,295 ns (0xFFFFFFFF)]	
NEXT	P	N/A		
RANGE	P	N/A	MIN	N/A
			MAX	N/A
			RES	N/A
			VLIST	N/A
Field	**Value**			
CS	XX_LATENCY_CONTROL (where XX shall be replaced by the appropriate two-letter abbreviation for the particular Entity)			
OCN:ICN:IPN	0:0:0			
Parameter Block	Layout 3 (See Section 5.3.2.6.3, "Layout 3 DataPart")			

5.3.3.2 TERMINAL COMMAND

This Command is used to manipulate the AudioControls inside a Terminal of the Audio Function. The exact layout of the Command is defined in Section 5.3.2.5, "AudioControl Command DataPart Layout."

The following paragraphs present a detailed description of all possible AudioControls a Terminal may incorporate. For each AudioControl, the supported Attributes and their value ranges are specified. Also, the appropriate AudioControl Selector value and the layout type of the Parameter Blocks are listed. The Control Selector codes are defined in Appendix A.23.1, "Terminal Control Selectors."

5.3.3.2.1 CLUSTER CONTROL

This AudioControl shall only be present in an Input Terminal that does not represent an AudioStreaming OUT Interface. All Output Terminals and any Input Terminal that represents an AudioStreaming OUT Interface shall not include this AudioControl. It shall always be implemented as Read (r) and shall never return a value of zero. See Section 5.3.3.1.2, "Cluster Control" for a detailed description. XX shall be replaced by TE.

5.3.3.2.2 CLUSTER ACTIVE CONTROL

This AudioControl shall always be present in an Input Terminal. An Output Terminal shall not include this AudioControl. See Section 5.3.3.1.3, "Cluster Active Control" for a detailed description. XX shall be replaced by TE.

5.3.3.2.3 VOLTAGE CONTROL

This AudioControl shall only be present in an Input Terminal that represents a microphone that requires either a phantom power supply or a bias voltage to properly operate. The Voltage Control is used to set the voltage level of the power supply.

Table 5-23: Voltage Control Characteristics

Voltage	SL	RW	OPT	
CUR	M	RoW		
NEXT	O	mRW		
RANGE	M	RO	MIN	[0 V (0x0000) to +255.9961 V (0xFFFF)]
			MAX	[0 V (0x0000) to +255.9961 V (0xFFFF)]
			RES	[0 V (0x0000) to +255.9961 V (0xFFFF)]
			VLIST	[0 V (0x0000) to +255.9961 V (0xFFFF)]+
Field	**Value**			
CS	TE_VOLTAGE_CONTROL			
OCN:ICN:IPN	The Pin channel number on which the AudioControl resides (including the Primary channel 0):0:0			
Parameter Block	Layout 2 (See Section 5.3.2.6.2, “Layout 2 DataPart”)			

5.3.3.2.4 MOMENTARY EXPOSURE LEVEL CONTROL

The Momentary Exposure Level Control is used to report the Momentary Exposure Level (MEL) for headphones, or a headphone jack as defined by EN 50332-3:2017, a CENELEC standard that defines a methodology for measuring the maximum Sound Pressure Level (SPL) experienced by a person listening to headphones. Consequently, only Output Terminals, representing headphones shall be permitted to have this MEL Control. If implemented, this AudioControl shall be Read-Only and shall have only the CUR attribute. Responding to the Pull Command returns the CUR attribute. The value of the MEL Control CUR attribute may range from 0.00 dB(A) to +255.9961 dB(A) (0xFFFF) in steps of 1/256 dB(A) or 0.00390625 dB(A) (0x0001).

The Control Selector field shall be set to TE_MEL_CONTROL and the Channel Number field shall be set to zero (Cluster-wide AudioControl). The parameter block for this AudioControl Command uses Layout 2 (See Section 5.2.1.3.2, “Layout 2 Parameter Block.”)

If the Device implements this AudioControl, it shall expose an interrupt endpoint in its AudioControl Interface as this AudioControl relies on interrupt endpoint communication to deliver a new MEL value every second. Old values will be overwritten if not read within the one second period. Polling the CUR attribute of this AudioControl is not a viable alternative and will lead to unreliable results since the availability of a new value solely relies on a 1 s timer maintained by the Device which could potentially be asynchronous to any clocks to which the Host may have access.

Table 5-24: Momentary Exposure Level Control Characteristics

Voltage	SL	RW	OPT	
CUR	M	RO	[0 dB (0x0000) to +255.9961 dB (0xFFFF)]	
NEXT	O	N/A		
RANGE	P	N/A	MIN	N/A
			MAX	N/A
			RES	N/A
			VLIST	N/A
Field	**Value**			
CS	TE_MOMENTARY_EXPOSURE_LEVEL_CONTROL			
OCN:ICN:IPN	0:0:0			
Parameter Block	Layout 2 (See Section 5.3.2.6.2, “Layout 2 DataPart”)			

5.3.3.2.5 OVERLOAD CONTROL

The Overload Control is used to indicate the existence of a thermal overload condition within a Terminal. (E.g., thermal overload in powered speakers; an amplifier, etc.)

Table 5-25: Overload Control Characteristics

Overload	SL	RW	OPT	
CUR	M	RO	[0 (0x00) to 1 (0x01)]; Normal or Overload Condition respectively.	
NEXT	P	N/A		
RANGE	I	N/A	MIN	N/A
			MAX	N/A
			RES	N/A
			VLIST	N/A
Field	**Value**			
CS	TE_ OVERLOAD_CONTROL			
OCN:ICN:IPN	0:0:0			
Parameter Block	Layout 1 (See Section 5.3.2.6.1, “Layout 1 DataPart”)			

5.3.3.2.6 SIGNAL CLIPPING CONTROL

The Signal Clipping Control is used to indicate the existence of signal clipping condition within a Terminal, i.e., a signal level exists outside of the (analog or digital) range that can be handled by the signal path.

Table 5-26: Clipping Control Characteristics

Signal Clipping	SL	RW	OPT	
CUR	M	RO	[0 (0x00) to 1 (0x01)]; Normal or Clipping Condition respectively.	
NEXT	P	N/A		
RANGE	I	N/A	MIN	N/A
			MAX	N/A
			RES	N/A
			VLIST	N/A
Field	**Value**			
CS	TE_CLIPPING_CONTROL			
OCN:ICN:IPN	0:0:0			
Parameter Block	Layout 1 (See Section 5.3.2.6.1, “Layout 1 DataPart”)			

5.3.3.2.7 LATENCY CONTROL

See Section 5.3.3.1.4, “Latency Control” for a detailed description. XX shall be replaced by TE.

5.3.3.3 MIXER UNIT COMMAND

This Command is used to manipulate the AudioControls inside the Mixer Unit. The exact layout of the Command is defined in Section 5.3.2.5, “AudioControl Command DataPart Layout.”

The following paragraphs present a detailed description of all possible AudioControls the Mixer Unit may incorporate. For each AudioControl, the supported Attributes and their value ranges are specified. Also, the appropriate Control Selector value and the layout type of the Parameter Blocks are listed. The Control Selector codes are defined in Appendix A.23.2, “Mixer Unit Control Selectors.”

5.3.3.3.1 MIXER CONTROL

The Mixer Control is used to manipulate the level at which an input channel of the Mixer Unit is mixed into one of its output channels.

Silence (i.e., -∞ dB) is represented by code 0x8000. If the AudioControl supports this setting, it shall explicitly indicate this by either advertising the subrange triplet [MIN: 0x8000; MAX: 0x8000; RES: 0x0000] or as the value 0x8000 being part of the VLIST enumeration.

Table 5-27: Mixer Control Characteristics

Mixer	SL	RW	MAN	
MCUR	M	RoW		
MNEXT	O	mRW		
MRANGE	M	RO	MIN	[-∞ dB (0x8000), -127.9961 dB (0x8001) to +127.9961 dB (0x7FFF)]
			MAX	[-∞ dB (0x8000), -127.9961 dB (0x8001) to +127.9961 dB (0x7FFF)]
			RES	[0 dB (0x0000) to +255.9961 dB (0xFFFF)]
			VLIST	[-∞ dB (0x8000), -127.9961 dB (0x8001) to +127.9961 dB (0x7FFF)]+
Field	**Value**			
CS	MU_MIXER_CONTROL			
OCN:ICN:IPN	As applicable			
Parameter Block	Layout 2 (See Section 5.3.2.6.2, "Layout 2 DataPart")			

5.3.3.3.2 CLUSTER CONTROL

See Section 5.3.3.1.2, "Cluster Control" for a detailed description. XX shall be replaced by MU.

5.3.3.3.3 CLUSTER ACTIVE CONTROL

This AudioControl shall always be present. See Section 5.3.3.1.3, "Cluster Active Control" for a detailed description. XX shall be replaced by TE.

5.3.3.3.4 LATENCY CONTROL

See Section 5.3.3.1.4, "Latency Control" for a detailed description. XX shall be replaced by MU.

5.3.3.4 SELECTOR UNIT COMMAND

This Command is used to manipulate the AudioControls inside a Selector Unit. The exact layout of the Command is defined in Section 5.3.2.5, "AudioControl Command DataPart Layout."

The following paragraphs present a detailed description of all possible AudioControls the Selector Unit may incorporate. For each AudioControl, the supported Attributes and their value ranges are specified. Also, the appropriate AudioControl Selector value and the layout type of the Parameter Blocks are listed. The Control Selector codes are defined in Appendix A.23.3, "Selector Unit Control Selectors."

5.3.3.4.1 SELECTOR CONTROL

The Selector Control provides the capability to select a particular Cluster from several incoming Clusters on the Selector Unit's Input Pins. The value zero, if available, is used to disconnect the output from all the inputs. In this case, the output Cluster is Inactive. The range for this AudioControl is from zero (disconnect option supported) or one (disconnect option not supported) to the number of Input Pins the Selector Unit supports, as reflected in the Entity's **bNrInputPins** Descriptor field.

Table 5-28: Selector Control Characteristics

Selector	SL	RW	MAN	
CUR	M	RoW	[0 to **bNrInputPins**]	
NEXT	O	mRW		
RANGE	M	RoW	MIN	0 or 1
			MAX	**bNrInputPins**
			RES	1
			VLIST	N/A
Field	**Value**			
CS	SU_SELECTOR_CONTROL			
OCN:ICN:IPN	0:0:0			
Parameter Block	Layout 1 (See Section 5.3.2.6.1, “Layout 1 DataPart”)			

5.3.3.4.2 LATENCY CONTROL

See Section 5.3.3.1.4, “Latency Control” for a detailed description. XX shall be replaced by SU.

5.3.3.5 FEATURE UNIT COMMAND

This Command is used to manipulate the AudioControls inside the Feature Unit. The exact layout of the Command is defined in Section 5.3.2.5, “AudioControl Command DataPart Layout.”

The following paragraphs present a detailed description of all possible AudioControls the Feature Unit may incorporate. For each AudioControl, the supported Attributes and their value ranges are specified. Also, the appropriate AudioControl Selector value and the layout type of the Parameter Blocks are listed. The Control Selector codes are defined in Appendix A.23.4, “Feature Unit Control Selectors.”

5.3.3.5.1.1 BYPASS CONTROL

See Section 5.3.3.1.1, “Bypass Control” for a detailed description. XX shall be replaced by FU.

5.3.3.5.2 MUTE CONTROL

The Mute Control is one of the building blocks of the Feature Unit. It is used to temporarily suppress all audio on an outgoing channel.

Table 5-29: Mute Control Characteristics

Mute	SL	RW	OPT	
CUR	M	RoW	[0 (0x00) to 1 (0x01)]; Unmuted or Muted respectively	
NEXT	O	mRW		
RANGE	I	N/A	MIN	N/A
			MAX	N/A
			RES	N/A
			VLIST	N/A
Field	**Value**			
CS	FU_MUTE_CONTROL			
OCN:ICN:IPN	OCN = ICN = The Pin channel number on which the AudioControl resides (including the Primary channel 0); IPN = 1			
Parameter Block	Layout 1 (See Section 5.3.2.6.1, "Layout 1 DataPart")			

5.3.3.5.3 GAIN CONTROL

The Gain Control is one of the building blocks of the Feature Unit. It is used to scale (amplify or attenuate) the audio level on a channel in the outgoing Cluster of the Feature Unit.

A Gain Control can be used to control volume for output signal paths and level for input signal paths.

Silence (i.e., -∞ dB) is represented by code 0x8000. If the AudioControl supports this setting, it shall explicitly indicate this by either advertising the subrange triplet [MIN: 0x8000; MAX: 0x8000; RES: 0x0000] or as the value 0x8000 being part of the VLIST enumeration.

Table 5-30: Gain Control Characteristics

Gain	SL	RW	OPT	
CUR	M	RoW		
NEXT	O	mRW		
RANGE	M	RO	MIN	[-∞ dB (0x8000), -127.9961 dB (0x8001) to +127.9961 dB (0x7FFF)]
			MAX	[-∞ dB (0x8000), -127.9961 dB (0x8001) to +127.9961 dB (0x7FFF)]
			RES	[0 dB (0x0000) to +255.9961 dB (0xFFFF)]
			VLIST	[-∞ dB (0x8000), -127.9961 dB (0x8001) to +127.9961 dB (0x7FFF)]+
Field	**Value**			
CS	FU_GAIN_CONTROL			
OCN:ICN:IPN	OCN = ICN = The Pin channel number on which the AudioControl resides (including the Primary channel 0); IPN = 1			
Parameter Block	Layout 2 (See Section 5.3.2.6.2, "Layout 2 DataPart")			

When the Calibrated bit D3 in the CAP Attribute is set on a microphone signal path's Gain Control, it indicates that the implementation has been calibrated so that, when the Gain Control is set to 0 dB, an eminently useful signal level will be produced at the AudioStreaming Interface when an average user positions the USB Audio Device in the manufacturer's intended manner and speaks at a conversational level. It is highly recommended that the power-up value for the Gain Control be set to 0 dB when the Calibrated bit is set.

In specific engineering terms, the rms level at the AudioStreaming Interface will measure -26 dBFS when the USB Audio Device is positioned in the manufacturer's intended manner with respect to a Head and Torso Simulator

(HATS) with its mouth simulator producing a superwideband (50 - 14,000 Hz) pink noise stimulus signal at a nominal level (89 dBSPL), as defined in IEEE Std 269-2019, "IEEE Standard for Measuring Electroacoustic Performance of Communication Devices".

Note that there are two common definitions of dBFS and they differ by 3 dB. This document uses the definition found within IEEE Std 269-2019.

- dBFS: Decibels are relative to full-scale digital saturation. 0 dBFS is the level of a square-wave signal whose amplitude is at the maximum digital value.

If the microphone signal path happens to have more than one Gain Control in series, each of the Gain Controls must have the Calibrated bit in its CAP Attribute set for the calibration to be meaningful.

If the USB Audio Device has any other audio signal processing on the microphone signal path that can be adjusted by the Host driver, the implementation's mic gain calibration shall be performed with those signal processing parameters set at nominal values, which are very likely the values the manufacturer decided to have those parameters take on after initial power-up.

5.3.3.5.4 BASS CONTROL

The Bass Control is one of the building blocks of the Feature Unit. It influences the general lower frequency behavior on an outgoing channel. Other parameters that also influence the lower frequency behavior, such as cut-off frequency, cannot be altered through this Command.

Table 5-31: Bass Control Characteristics

Bass	SL	RW	OPT	
CUR	M	RoW		
NEXT	O	mRW		
RANGE	M	RO	MIN	[-128.0000 dB (0x8000) to +127.9961 dB (0x7FFF)]
			MAX	[-128.0000 dB (0x8000) to +127.9961 dB (0x7FFF)]
			RES	[0 dB (0x0000) to +255.9961 dB (0xFFFF)]
			VLIST	[-128.0000 dB (0x8001) to +127.9961 dB (0x7FFF)]+
Field	**Value**			
CS	FU_BASS_CONTROL			
OCN:ICN:IPN	OCN = ICN = The Pin channel number on which the AudioControl resides (including the Primary channel 0); IPN = 1			
Parameter Block	Layout 2 (See Section 5.3.2.6.2, "Layout 2 DataPart")			

5.3.3.5.5 MID CONTROL

The Mid Control is one of the building blocks of the Feature Unit. It influences the general medium frequency behavior on an outgoing channel. Other parameters that also influence the medium frequency behavior, such as cut-off frequencies, cannot be altered through this Command.

Table 5-32: Mid Control Characteristics

Mid	SL	RW	OPT	
CUR	M	RoW		
NEXT	O	mRW		
RANGE	M	RO	MIN	[-128.0000 dB (0x8000) to +127.9961 dB (0x7FFF)]
			MAX	[-128.0000 dB (0x8000) to +127.9961 dB (0x7FFF)]
			RES	[0 dB (0x0000) to +255.9961 dB (0xFFFF)]
			VLIST	[-128.0000 dB (0x8001) to +127.9961 dB (0x7FFF)]+
Field	**Value**			
CS	FU_MID_CONTROL			
OCN:ICN:IPN	OCN = ICN = The Pin channel number on which the AudioControl resides (including the Primary channel 0); IPN = 1			
Parameter Block	Layout 2 (See Section 5.3.2.6.2, "Layout 2 DataPart")			

5.3.3.5.6 TREBLE CONTROL

The Treble Control is one of the building blocks of the Feature Unit. It influences the general higher frequency behavior on an outgoing channel. Other parameters that also influence the higher frequency behavior, such as cut-off frequency, cannot be altered through this Command.

Table 5-33: Treble Control Characteristics

Treble	SL	RW	OPT	
CUR	M	RoW		
NEXT	O	mRW		
RANGE	M	RO	MIN	[-128.0000 dB (0x8000) to +127.9961 dB (0x7FFF)]
			MAX	[-128.0000 dB (0x8000) to +127.9961 dB (0x7FFF)]
			RES	[0 dB (0x0000) to +255.9961 dB (0xFFFF)]
			VLIST	[-128.0000 dB (0x8001) to +127.9961 dB (0x7FFF)]+
Field	**Value**			
CS	FU_TREBLE_CONTROL			
OCN:ICN:IPN	OCN = ICN = The Pin channel number on which the AudioControl resides (including the Primary channel 0); IPN = 1			
Parameter Block	Layout 2 (See Section 5.3.2.6.2, "Layout 2 DataPart")			

5.3.3.5.7 GRAPHIC EQUALIZER CONTROL

The Graphic Equalizer Control is one of the optional building blocks of the Feature Unit. The Audio Device Class definition provides for standard support of a third octave graphic equalizer. The bands are defined according to the ANSI S1.11-2004 standard. Bands are numbered from 14 (midband frequency of 25 Hz) up to 43 (midband frequency of 20,000 Hz), making a total of 30 possible bands. The following table lists the band numbers and their midband frequencies

Table 5-34: Band Numbers and Nominal Midband Frequencies (ANSI S1.11-2004 Standard)

Band Nr.	Midband Freq.	Band Nr.	Midband Freq.	Band Nr.	Midband Freq.
14	25 Hz	24*	250 Hz	34	2500 Hz

Band Nr.	Midband Freq.	Band Nr.	Midband Freq.	Band Nr.	Midband Freq.
15*	31.5 Hz	25	315 Hz	35	3150 Hz
16	40 Hz	26	400 Hz	36*	4000 Hz
17	50 Hz	27*	500 Hz	37	5000 Hz
18*	63 Hz	28	630 Hz	38	6300 Hz
19	80 Hz	29	800 Hz	39*	8000 Hz
20	100 Hz	30*	1000 Hz	40	10000 Hz
21*	125 Hz	31	1250 Hz	41	12500 Hz
22	160 Hz	32	1600 Hz	42*	16000 Hz
23	200 Hz	33*	2000 Hz	43	20000 Hz

Note: Bands marked with an asterisk (*) are those present in an octave equalizer.

A Feature Unit that supports the Graphic Equalizer Control is not required to implement the full set of filters. A subset (for example, octave bands) may be implemented. During a Pull Command, the **bmBandsPresent** field in the Parameter Block is a bitmap indicating which bands are effectively implemented and thus reported back in the returned Parameter Block. Consequently, the number of bits set in this field determines the total length of the returned Parameter Block. During a Push Command, a bit set in the **bmBandsPresent** field indicates there is a new setting for that band in the Parameter Block that follows. The new values shall be in ascending order. If the number of bits set in the **bmBandsPresent** field does not match the number of parameters specified in the following block, the Command Set Request shall return a Request Error.

The Parameter Block for the CUR and NEXT Attributes of the Graphic Equalizer Control is as follows:

Table 5-35: Graphic Equalizer CUR or NEXT Control Characteristics

Graphic Equalizer	SL	RW	OPT		
CUR	M	RoW			
NEXT	O	mRW			
VALUE LIST	N/A				
Field	Value				
CS	FU_GRAPHIC_EQUALIZER_CONTROL				
OCN:ICN:IPN	OCN = ICN = The Pin channel number on which the AudioControl resides (including the Primary channel 0); IPN = 1				
Parameter Block	Length	4 + (2 * number of bits set in **bmBandsPresent**: 2 * NrBits)			
Offset	Field		Size	Value	Description
0	bmBandsPresent		4	Bitmap	A bit set indicates the band is present: D0: Band 14 is present. D1: Band 15 is present. … D29: Band 43 is present. D30: Reserved. D31: Reserved.
4	wCUR(Lowest)		2	Number	The setting for the CUR Attribute of the lowest band present: [-127.9961 dB (0x8001) to +127.9961 dB (0x7FFF)]
…	…		…	…	…
4+2*(NrBits-1)	wCUR(Highest)		2	Number	The setting for the CUR Attribute of the highest band present.

The Parameter Block for the RANGE Attribute of the Graphic Equalizer Control is as follows:

Table 5-36: Graphic Equalizer RANGE Control Characteristics

Graphic Equalizer	SL	RW	OPT		
RANGE	M	RO			
VALUE LIST	N/A				
Field	Value				
CS	FU_GRAPHIC_EQUALIZER_CONTROL				
OCN:ICN:IPN	OCN = ICN = The Pin channel number on which the AudioControl resides (including the Primary channel 0); IPN = 1				
Parameter Block	Length	2+6*n			
Offset	Field		Size	Value	Description
0	wNumSubRanges		2	Number	The number of subranges of the Graphic Equalizer Control: n
2	wMIN(1)		2	Number	The setting for the MIN Attribute of the first subrange of the Graphic Equalizer Control
4	wMAX(1)		2	Number	The setting for the MAX Attribute of the first subrange of the Graphic Equalizer Control
6	wRES(1)		2	Number	The setting for the RES Attribute of the first subrange of the Graphic Equalizer Control
…	…		…	…	…
2+6*(n-1)	wMIN(n)		1	Number	The setting for the MIN Attribute of the last subrange of the Graphic Equalizer Control
4+6*(n-1)	wMAX(n)		1	Number	The setting for the MAX Attribute of the last subrange of the Graphic Equalizer Control
6+6*(n-1)	wRES(n)		1	Number	The setting for the RES Attribute of the last subrange of the Graphic Equalizer Control

5.3.3.5.8 AUTOMATIC GAIN CONTROL

The Automatic Gain Control (AGC) is one of the building blocks of the Feature Unit. It enables or disables the circuitry that tries to optimize the gain setting automatically and continuously on an outgoing channel. The algorithm that achieves this optimization is not defined by this specification and is implementation dependent.

Table 5-37: Automatic Gain Control Characteristics

Automatic Gain	SL	RW	OPT	
CUR	M	RoW	[0 (0x00) to 1 (0x01)]; Automatic Gain Disabled or Enabled respectively.	
NEXT	O	mRW		
RANGE	I	N/A	MIN	N/A
			MAX	N/A
			RES	N/A
			VLIST	N/A
Field	**Value**			
CS	FU_AUTOMATIC_GAIN_CONTROL			
OCN:ICN:IPN	OCN = ICN = The Pin channel number on which the AudioControl resides (including the Primary channel 0); IPN = 1			
Parameter Block	Layout 1 (See Section 5.3.2.6.1, “Layout 1 DataPart”)			

5.3.3.5.9 DELAY CONTROL

The Delay Control is one of the building blocks of the Feature Unit. It impacts the amount of delay that is incurred on an outgoing channel.

Table 5-38: Delay Control Characteristics

Delay	SL	RW	OPT	
CUR	M	RoW		
NEXT	O	mRW		
RANGE	M	RO	MIN	[0 s (0x00000000) to 1023.999999761581 s (0xFFFFFFFF)]
			MAX	[0 s (0x00000000) to 1023.999999761581 s (0xFFFFFFFF)]
			RES	[0 s (0x00000000) to 1023.999999761581 s (0xFFFFFFFF)]
			VLIST	[0 s (0x00000000) to 1023.999999761581 s (0xFFFFFFFF)]+
Field	**Value**			
CS	FU_DELAY_CONTROL			
OCN:ICN:IPN	OCN = ICN = The Pin channel number on which the AudioControl resides (including the Primary channel 0); IPN = 1			
Parameter Block	Layout 3 (See Section 5.3.2.6.3, “Layout 3 DataPart”)			

5.3.3.5.10 BASS BOOST CONTROL

The Bass Boost Control is one of the building blocks of the Feature Unit. It enables or disables the circuitry that boosts the lower frequency content on an outgoing channel. The algorithm that achieves this boost is not defined by this specification and is implementation dependent.

Table 5-39: Bass Boost Control Characteristics

Bass Boost	SL	RW	OPT	
CUR	M	RoW	[0 (0x00) to 1 (0x01)]; Bass Boost Disabled or Enabled respectively.	
NEXT	O	mRW		
RANGE	I	N/A	MIN	N/A
			MAX	N/A
			RES	N/A
			VLIST	N/A
Field	**Value**			
CS	FU_BASS_BOOST_CONTROL			
OCN:ICN:IPN	OCN = ICN = The Pin channel number on which the AudioControl resides (including the Primary channel 0); IPN = 1			
Parameter Block	Layout 1 (See Section 5.3.2.6.1, “Layout 1 DataPart”)			

5.3.3.5.11 LOUDNESS CONTROL

The Loudness Control is one of the building blocks of the Feature Unit. It enables or disables the circuitry that boosts the bass content at lower volumes on an outgoing channel. The algorithm that achieves this boost is not defined by this specification and is implementation dependent.

Table 5-40: Loudness Control Characteristics

Loudness	SL	RW	OPT	
CUR	M	RoW	[0 (0x00) to 1 (0x01)]; Loudness Disabled or Enabled respectively.	
NEXT	O	mRW		
RANGE	I	N/A	MIN	N/A
			MAX	N/A
			RES	N/A
			VLIST	N/A
Field	**Value**			
CS	FU_LOUDNESS_CONTROL			
OCN:ICN:IPN	OCN = ICN = The Pin channel number on which the AudioControl resides (including the Primary channel 0); IPN = 1			
Parameter Block	Layout 1 (See Section 5.3.2.6.1, “Layout 1 DataPart”)			

5.3.3.5.12 INPUT GAIN PAD CONTROL

The Input Gain Pad Control is one of the building blocks of the Feature Unit. It is used to scale (amplify or attenuate) the audio level on a channel in the outgoing Cluster of the Feature Unit, typically in a few coarse steps.

Silence (i.e., -∞ dB) is represented by code 0x8000. If the AudioControl supports this setting, it shall explicitly indicate this by either advertising the subrange triplet [MIN: 0x8000; MAX: 0x8000; RES: 0x0000] or as the value 0x8000 being part of the VLIST enumeration.

Table 5-41: Input Gain Pad Control Characteristics

Input Gain Pad	SL	RW	OPT	
CUR	M	RoW		
NEXT	O	mRW		
RANGE	M	RO	MIN	[-∞ dB (0x8000), -127.9961 dB (0x8001) to +127.9961 dB (0x7FFF)]
			MAX	[-∞ dB (0x8000), -127.9961 dB (0x8001) to +127.9961 dB (0x7FFF)]
			RES	[0 dB (0x0000) to +255.9961 dB (0xFFFF)]
			VLIST	[-∞ dB (0x8000), -127.9961 dB (0x8001) to +127.9961 dB (0x7FFF)]+
Field	**Value**			
CS	FU_INPUT_GAIN_PAD_CONTROL			
OCN:ICN:IPN	OCN = ICN = The Pin channel number on which the AudioControl resides (including the Primary channel 0); IPN = 1			
Parameter Block	Layout 2 (See Section 5.3.2.6.2, "Layout 2 DataPart")			

5.3.3.5.13 PHASE INVERTER CONTROL

The Phase Inverter Control is one of the building blocks of the Feature Unit. It inverts (180°phase shift) the signal on an outgoing channel.

Table 5-42: Phase Inverter Control Characteristics

Phase Inverter	SL	RW	OPT	
CUR	M	RoW	[0 (0x00) to 1 (0x01)]; Phase Inverter Disabled or Enabled respectively.	
NEXT	O	mRW		
RANGE	I	N/A	MIN	N/A
			MAX	N/A
			RES	N/A
			VLIST	N/A
Field	**Value**			
CS	FU_PHASE_INVERTER_CONTROL			
OCN:ICN:IPN	OCN = ICN = The Pin channel number on which the AudioControl resides (including the Primary channel 0); IPN = 1			
Parameter Block	Layout 1 (See Section 5.3.2.6.1, "Layout 1 DataPart")			

5.3.3.5.14 LATENCY CONTROL

See Section 5.3.3.1.4, "Latency Control" for a detailed description. XX shall be replaced by FU.

5.3.3.6 SAMPLE RATE CONVERTER UNIT COMMAND

This Command is used to manipulate the AudioControls inside the SRC Unit. The exact layout of the Command is defined in Section 5.3.2.5, "AudioControl Command DataPart Layout."

The following paragraphs present a detailed description of all possible AudioControls the SRC Unit may incorporate. For each AudioControl, the supported Attributes and their value ranges are specified. Also, the appropriate AudioControl Selector value and the layout type of the Parameter Blocks are listed. The Control Selector codes are defined in Appendix A.23.5, "SRC Unit Control Selectors."

5.3.3.6.1 LATENCY CONTROL

See Section 5.3.3.1.4, “Latency Control” for a detailed description. XX shall be replaced by RU.

5.3.3.7 EFFECT UNIT COMMAND

This Command is used to manipulate the AudioControls inside Effect Unit. The exact layout of the Command is defined in Section 5.3.2.5, “AudioControl Command DataPart Layout.”

The following paragraphs present a detailed description of all possible AudioControls Effect Unit may incorporate. For each AudioControl, the supported Attributes and their value ranges are specified. Also, the appropriate AudioControl Selector value and the layout type of the Parameter Blocks are listed. The Control Selector codes are defined in Appendix A.23.6, “Effect Unit Control Selectors.”

5.3.3.7.1 PARAMETRIC EQUALIZER SECTION EFFECT UNIT

At least one of the optional AudioControls shall be implemented.

5.3.3.7.1.1 BYPASS CONTROL

See Section 5.3.3.1.1, “Bypass Control” for a detailed description. XX shall be replaced by PE.

5.3.3.7.1.2 CENTER FREQUENCY CONTROL

The Center Frequency Control is used to manipulate the center frequency of the filter implemented on an outgoing channel.

Table 5-43: Center Frequency Control Characteristics

Center Frequency	SL	RW	OPT	
CUR	M	RoW		
NEXT	O	mRW		
RANGE	M	RO	MIN	[0 Hz (0x00000000) to 65,535.9999847 Hz (0xFFFFFFFF)]
			MAX	[0 Hz (0x00000000) to 65,535.9999847 Hz (0xFFFFFFFF)]
			RES	[0 Hz (0x00000000) to 65,535.9999847 Hz (0xFFFFFFFF)]
			VLIST	[0 Hz (0x00000000) to 65,535.9999847 Hz (0xFFFFFFFF)]+
Field	**Value**			
CS	PE_CENTER_FREQUENCY_CONTROL			
OCN:ICN:IPN	OCN = ICN = The Pin channel number on which the AudioControl resides (including the Primary channel 0); IPN = 1			
Parameter Block	Layout 3 (See Section 5.3.2.6.3, “Layout 3 DataPart”)			

5.3.3.7.1.3 QFACTOR CONTROL

The Qfactor Control is used to manipulate the range of frequencies affected around the center frequency of the filter implemented on an outgoing channel.

Note: The Q-factor of a filter is defined as the ratio of the center frequency to the bandwidth measured at the -3 dB point. The result of a Q setting of 10 for a filter set to 1000 Hz is a bandwidth of 100 Hz. Likewise, a center frequency of 5325 Hz and a Q setting of 7.25 results in a bandwidth of 734.48275862 Hz.

Table 5-44: Qfactor Control Characteristics

Qfactor	SL	RW	OPT	
CUR	M	RoW		
NEXT	O	mRW		
RANGE	M	RO	MIN	[0 (0x00000000) to 65,535.9999847 (0xFFFFFFFF)]
			MAX	[0 (0x00000000) to 65,535.9999847 (0xFFFFFFFF)]
			RES	[0 (0x00000000) to 65,535.9999847 (0xFFFFFFFF)]
			VLIST	[0 (0x00000000) to 65,535.9999847 (0xFFFFFFFF)]+
Field	**Value**			
CS	PE_QFACTOR_CONTROL			
OCN:ICN:IPN	OCN = ICN = The Pin channel number on which the AudioControl resides (including the Primary channel 0); IPN = 1			
Parameter Block	Layout 3 (See Section 5.3.2.6.3, “Layout 3 DataPart”)			

5.3.3.7.1.4 GAIN CONTROL

The Gain Control is used to manipulate the amount of gain or attenuation at the center frequency of the filter implemented on an outgoing channel.

Table 5-45: Gain Control Characteristics

Gain	SL	RW	OPT	
CUR	M	RoW		
NEXT	O	mRW		
RANGE	M	RO	MIN	[-128.0000 dB (0x8000) to +127.9961 dB (0x7FFF)]
			MAX	[-128.0000 dB (0x8000) to +127.9961 dB (0x7FFF)]
			RES	[0 dB (0x0000) to +255.9961 dB (0xFFFF)]
			VLIST	[-128.0000 dB (0x8000) to +127.9961 dB (0x7FFF)]+
Field	**Value**			
CS	PE_GAIN_CONTROL			
OCN:ICN:IPN	OCN = ICN = The Pin channel number on which the AudioControl resides (including the Primary channel 0); IPN = 1			
Parameter Block	Layout 2 (See Section 5.3.2.6.2, “Layout 2 DataPart”)			

5.3.3.7.1.5 LATENCY CONTROL

See Section 5.3.3.1.4, “Latency Control” for a detailed description. XX shall be replaced by PE.

5.3.3.7.2 REVERBERATION EFFECT UNIT

At least one of the optional AudioControls shall be implemented.

5.3.3.7.2.1 BYPASS CONTROL

See Section 5.3.3.1.1, “Bypass Control” for a detailed description. XX shall be replaced by RV.

5.3.3.7.2.2 TYPE CONTROL

The Type Control is a macro AudioControl that selects among potentially different algorithms and internal parameter settings to obtain the desired reverberation effect on an outgoing channel. The CUR and NEXT range for this AudioControl is from one to the number of supported Reverberation Types, as reflected in the Entity's **bNrTypes** Descriptor field.

Table 5-46: Type Control Characteristics

Type	SL	RW	OPT	
CUR	M	RoW	[1 to **bNrTypes**]	
NEXT	O	mRW	[1 to **bNrTypes**]	
RANGE	P	N/A	MIN	N/A
			MAX	N/A
			RES	N/A
			VLIST	N/A
Field	**Value**			
CS	RV_TYPE_CONTROL			
OCN:ICN:IPN	OCN = ICN = The Pin channel number on which the AudioControl resides (including the Primary channel 0); IPN = 1			
Parameter Block	Layout 1 (See Section 5.3.2.6.1, "Layout 1 DataPart")			

5.3.3.7.2.3 LEVEL CONTROL

The Level Control is used to set the amount of reverberant sound introduced on an outgoing channel, expressed as a percentage, compared to the level of the original signal.

Table 5-47: Level Control Characteristics

Level	SL	RW	OPT	
CUR	M	RoW		
NEXT	O	mRW		
RANGE	M	RO	MIN	[0 % (0x00) to 255 % (0xFF)]
			MAX	[0 % (0x00) to 255 % (0xFF)]
			RES	[0 % (0x00) to 255 % (0xFF)]
			VLIST	[0 % (0x00) to 255 % (0xFF)]+
Field	**Value**			
CS	RV_LEVEL_CONTROL			
OCN:ICN:IPN	OCN = ICN = The Pin channel number on which the AudioControl resides (including the Primary channel 0); IPN = 1			
Parameter Block	Layout 1 (See Section 5.3.2.6.1, "Layout 1 DataPart")			

5.3.3.7.2.4 TIME CONTROL

The Time Control is used to set the time over which the reverberation, introduced on an outgoing channel, will continue

Table 5-48: Time Control Characteristics

Time	SL	RW	OPT	
CUR	M	RoW		
NEXT	O	mRW		
RANGE	M	RO	MIN	[0 s (0x0000) to 255.9961 s (0xFFFF)]
			MAX	[0 s (0x0000) to 255.9961 s (0xFFFF)]
			RES	[0 s (0x0000) to 255.9961 s (0xFFFF)]
			VLIST	[0 s (0x0000) to 255.9961 s (0xFFFF)]+
Field	**Value**			
CS	RV_TIME_CONTROL			
OCN:ICN:IPN	OCN = ICN = The Pin channel number on which the AudioControl resides (including the Primary channel 0); IPN = 1			
Parameter Block	Layout 2 (See Section 5.3.2.6.2, “Layout 2 DataPart”)			

5.3.3.7.2.5 DELAY FEEDBACK CONTROL

The Delay Feedback Control is used when the reverb type is set to Type 6 (Delay) or Type 7 (Panning Delay). It is ignored for all other Reverb Types. It sets the way in which delay repeats on an outgoing channel.

Table 5-49: Feedback Control Characteristics

Feedback	SL	RW	OPT	
CUR	M	RoW		
NEXT	O	mRW		
RANGE	M	RO	MIN	[0 % (0x00) to 255 % (0xFF)]
			MAX	[0 % (0x00) to 255 % (0xFF)]
			RES	[0 % (0x00) to 255 % (0xFF)]
			VLIST	[0 % (0x00) to 255 % (0xFF)]+
Field	**Value**			
CS	RV_FEEDBACK_CONTROL			
OCN:ICN:IPN	OCN = ICN = The Pin channel number on which the AudioControl resides (including the Primary channel 0); IPN = 1			
Parameter Block	Layout 1 (See Section 5.3.2.6.1, “Layout 1 DataPart”)			

5.3.3.7.2.6 PRE-DELAY CONTROL

The Pre-Delay Control is used to set the delay time between the original source and the initial reflection in the reverberation, introduced on an outgoing channel. The Pre-Delay Control shall support the CUR and RANGE(MIN, MAX, RES) Attributes. The settings for the CUR, MIN, MAX, and RES Attributes may range from 0 ms (0x0000) to 65535 ms (0xFFFF) in steps of 1 ms (0x0001).

The Control Selector field shall be set to RV_PREDELAY_CONTROL and the Channel Number field indicates the desired Channel.

Table 5-50: Pre-Delay Control Characteristics

Pre-Delay	SL	RW	OPT	
CUR	M	RoW		
NEXT	O	mRW		
RANGE	M	RO	MIN	[0 ms (0x0000) to 65,535 ms (0xFFFF)]
			MAX	[0 ms (0x0000) to 65,535 ms (0xFFFF)]
			RES	[0 ms (0x0000) to 65,535 ms (0xFFFF)]
			VLIST	[0 ms (0x0000) to 65,535 ms (0xFFFF)]+
Field	**Value**			
CS	RV_PREDELAY_CONTROL			
OCN:ICN:IPN	OCN = ICN = The Pin channel number on which the AudioControl resides (including the Primary channel 0); IPN = 1			
Parameter Block	Layout 2 (See Section 5.3.2.6.2, “Layout 2 DataPart”)			

5.3.3.7.2.7 DENSITY CONTROL

The Density Control is used to set the density of reflections in the reverberant sound introduced on an outgoing channel.

Table 5-51: Density Control Characteristics

Density	SL	RW	OPT	
CUR	M	RoW		
NEXT	O	mRW		
RANGE	M	RO	MIN	[0 (0x00) to 100 % (0x64)]
			MAX	[0 (0x00) to 100 % (0x64)]
			RES	[0 (0x00) to 100 % (0x64)]
			VLIST	[0 (0x00) to 100 % (0x64)]+
Field	**Value**			
CS	RV_DENSITY_CONTROL			
OCN:ICN:IPN	OCN = ICN = The Pin channel number on which the AudioControl resides (including the Primary channel 0); IPN = 1			
Parameter Block	Layout 1 (See Section 5.3.2.6.1, “Layout 1 DataPart”)			

5.3.3.7.2.8 HI-FREQ ROLL-OFF CONTROL

The Hi-Freq Roll-Off Control is used to set the frequency of a low pass filter on the reverberant sound introduced on an outgoing channel.

Table 5-52: Hi-Freq Roll-Off Control Characteristics

Hi-Freq Roll-Off	SL	RW	OPT	
CUR	M	RoW		
NEXT	O	mRW		
RANGE	M	RO	MIN	[0 Hz (0x00000000) to 65,535.9999847 Hz (0xFFFFFFFF)]
			MAX	[0 Hz (0x00000000) to 65,535.9999847 Hz (0xFFFFFFFF)]
			RES	[0 Hz (0x00000000) to 65,535.9999847 Hz (0xFFFFFFFF)]
			VLIST	[0 Hz (0x00000000) to 65,535.9999847 Hz (0xFFFFFFFF)]+
Field	**Value**			
CS	PE_HIFREQ_ROLLOFF_CONTROL			
OCN:ICN:IPN	OCN = ICN = The Pin channel number on which the AudioControl resides (including the Primary channel 0); IPN = 1			
Parameter Block	Layout 3 (See Section 5.3.2.6.3, “Layout 3 DataPart”)			

5.3.3.7.2.9 LATENCY CONTROL

See Section 5.3.3.1.4, “Latency Control” for a detailed description. XX shall be replaced by RV.

5.3.3.7.3 MODULATION DELAY EFFECT UNIT

5.3.3.7.3.1 BYPASS CONTROL

See Section 5.3.3.1.1, “Bypass Control” for a detailed description. XX shall be replaced by MD.

5.3.3.7.3.2 BALANCE CONTROL

The Balance Control is used to set the amount of effect sound introduced on an outgoing channel. A setting of -100 % results in 100 % of original signal and 0 % of effected signal. A setting of 0 % results in equal amounts of original signal and effected signal. A setting of +100 % results in 0 % of original signal and 100 % of effected signal.

Table 5-53: Balance Control Characteristics

Balance	SL	RW	OPT	
CUR	M	RoW		
NEXT	O	mRW		
RANGE	M	RO	MIN	[-100 % (0x9C) to 100 % (0x64)]
			MAX	[-100 % (0x9C) to 100 % (0x64)]
			RES	[-100 % (0x9C) to 100 % (0x64)]
			VLIST	[-100 % (0x9C) to 100 % (0x64)]+
Field	**Value**			
CS	MD_BALANCE_CONTROL			
OCN:ICN:IPN	OCN = ICN = The Pin channel number on which the AudioControl resides (including the Primary channel 0); IPN = 1			
Parameter Block	Layout 1 (See Section 5.3.2.6.1, “Layout 1 DataPart”)			

5.3.3.7.3.3 RATE CONTROL

The Rate Control is used to set the speed (frequency) of the modulator of the delay time introduced on an outgoing channel. Higher values result in faster modulation.

Table 5-54: Rate Control Characteristics

Rate	SL	RW	OPT	
CUR	M	RoW		
NEXT	O	mRW		
RANGE	M	RO	MIN	[0 Hz (0x0000) to 255.9961 Hz (0xFFFF)]
			MAX	[0 Hz (0x0000) to 255.9961 Hz (0xFFFF)]
			RES	[0 Hz (0x0000) to 255.9961 Hz (0xFFFF)]
			VLIST	[0 Hz (0x0000) to 255.9961 Hz (0xFFFF)]+
Field	**Value**			
CS	MD_RATE_CONTROL			
OCN:ICN:IPN	OCN = ICN = The Pin channel number on which the AudioControl resides (including the Primary channel 0); IPN = 1			
Parameter Block	Layout 2 (See Section 5.3.2.6.2, “Layout 2 DataPart”)			

5.3.3.7.3.4 DEPTH CONTROL

The Depth Control is used to set the depth at which the effect sound introduced on an outgoing channel is modulated. Higher values result in deeper modulation.

Table 5-55: Depth Control Characteristics

Depth	SL	RW	OPT	
CUR	M	RoW		
NEXT	O	mRW		
RANGE	M	RO	MIN	[0 ms (0x0000) to 255.9961 ms (0xFFFF)]
			MAX	[0 ms (0x0000) to 255.9961 ms (0xFFFF)]
			RES	[0 ms (0x0000) to 255.9961 ms (0xFFFF)]
			VLIST	[0 ms (0x0000) to 255.9961 ms (0xFFFF)]
Field	**Value**			
CS	MD_DEPTH_CONTROL			
OCN:ICN:IPN	OCN = ICN = The Pin channel number on which the AudioControl resides (including the Primary channel 0); IPN = 1			
Parameter Block	Layout 2 (See Section 5.3.2.6.2, “Layout 2 DataPart”)			

5.3.3.7.3.5 TIME CONTROL

The Time Control is used to set the length of the delay time introduced on an outgoing channel. Higher values result in longer times.

Table 5-56: Time Control Characteristics

Time	SL	RW	OPT	
CUR	M	RoW		
NEXT	O	mRW		
RANGE	M	RO	MIN	[0 ms (0x0000) to 255.9961 ms (0xFFFF)]
			MAX	[0 ms (0x0000) to 255.9961 ms (0xFFFF)]
			RES	[0 ms (0x0000) to 255.9961 ms (0xFFFF)]
			VLIST	[0 ms (0x0000) to 255.9961 ms (0xFFFF)]+
Field	**Value**			
CS	MD_TIME_CONTROL			
OCN:ICN:IPN	OCN = ICN = The Pin channel number on which the AudioControl resides (including the Primary channel 0); IPN = 1			
Parameter Block	Layout 2 (See Section 5.3.2.6.2, “Layout 2 DataPart”)			

5.3.3.7.3.6 FEEDBACK LEVEL CONTROL

The Feedback Level Control is used to set the level at which the effected (delayed) sound introduced on an outgoing channel is mixed back into its own input. Higher values result in higher level of feedback, thereby resulting in more audible delay repeats.

Table 5-57: Feedback Control Characteristics

Feedback	SL	RW	OPT	
CUR	M	RoW		
NEXT	O	mRW		
RANGE	M	RO	MIN	[0 % (0x00) to 255 % (0xFF)]
			MAX	[0 % (0x00) to 255 % (0xFF)]
			RES	[0 % (0x00) to 255 % (0xFF)]
			VLIST	[0 % (0x00) to 255 % (0xFF)]+
Field	**Value**			
CS	MD_FEEDBACK_CONTROL			
OCN:ICN:IPN	OCN = ICN = The Pin channel number on which the AudioControl resides (including the Primary channel 0); IPN = 1			
Parameter Block	Layout 1 (See Section 5.3.2.6.1, “Layout 1 DataPart”)			

5.3.3.7.3.7 LATENCY CONTROL

See Section 5.3.3.1.4, “Latency Control” for a detailed description. XX shall be replaced by MD.

5.3.3.7.4 DYNAMIC RANGE COMPRESSOR/EXPANDER EFFECT UNIT

5.3.3.7.4.1 BYPASS CONTROL

See Section 5.3.3.1.1, “Bypass Control” for a detailed description. XX shall be replaced by DR.

5.3.3.7.4.2 RATIO CONTROL

The Ratio Control is used to influence the slope of the active part of the static input-to-output transfer characteristic of the DRC on an outgoing channel.

Table 5-58: Compression Ratio Control Characteristics

Compression Ratio	SL	RW	OPT	
CUR	M	RoW		
NEXT	O	mRW		
RANGE	M	RO	MIN	[0 (0x0000) to 255.9961 (0xFFFF)]
			MAX	[0 (0x0000) to 255.9961 (0xFFFF)]
			RES	[0 (0x0000) to 255.9961 (0xFFFF)]
			VLIST	[0 (0x0000) to 255.9961 (0xFFFF)]+
Field	**Value**			
CS	DR _RATIO_CONTROL			
OCN:ICN:IPN	OCN = ICN = The Pin channel number on which the AudioControl resides (including the Primary channel 0); IPN = 1			
Parameter Block	Layout 2 (See Section 5.3.2.6.2, “Layout 2 DataPart”)			

5.3.3.7.4.3 MAXAMPL CONTROL

The MaxAmpl Control is used to set the upper boundary of the active input range of the DRC on an outgoing channel.

Table 5-59: MaxAmpl Control Characteristics

MaxAmpl	SL	RW	OPT	
CUR	M	RoW		
NEXT	O	mRW		
RANGE	M	RO	MIN	[-128.0000 dB (0x8000) to +127.9961 dB (0x7FFF)]
			MAX	[-128.0000 dB (0x8000) to +127.9961 dB (0x7FFF)]
			RES	[0 dB (0x0000) to +255.9961 dB (0xFFFF)]
			VLIST	[-128.0000 dB (0x8000) to +127.9961 dB (0x7FFF)]+
Field	**Value**			
CS	DR_MAXAMPL_CONTROL			
OCN:ICN:IPN	OCN = ICN = The Pin channel number on which the AudioControl resides (including the Primary channel 0); IPN = 1			
Parameter Block	Layout 2 (See Section 5.3.2.6.2, “Layout 2 DataPart”)			

5.3.3.7.4.4 THRESHOLD CONTROL

The Threshold Control is used to set the lower boundary of the active input range of the effect on an outgoing channel.

Table 5-60: Threshold Control Characteristics

Threshold	SL	RW	OPT	
CUR	M	RoW		
NEXT	O	mRW		
RANGE	M	RO	MIN	[-128.0000 dB (0x8000) to +127.9961 dB (0x7FFF)]
			MAX	[-128.0000 dB (0x8000) to +127.9961 dB (0x7FFF)]
			RES	[0 dB (0x0000) to +255.9961 dB (0xFFFF)]
			VLIST	[-128.0000 dB (0x8000) to +127.9961 dB (0x7FFF)]+
Field	**Value**			
CS	DR_THRESHOLD_CONTROL			
OCN:ICN:IPN	OCN = ICN = The Pin channel number on which the AudioControl resides (including the Primary channel 0); IPN = 1			
Parameter Block	Layout 2 (See Section 5.3.2.6.2, “Layout 2 DataPart”)			

5.3.3.7.4.5 ATTACK TIME CONTROL

The Attack Time Control is used to determine the response of the DRC on an outgoing channel to a step increase in the signal level on the corresponding input channel.

Table 5-61: Attack Time Control Characteristics

Attack Time	SL	RW	OPT	
CUR	M	RoW		
NEXT	O	mRW		
RANGE	M	RO	MIN	[0 ms (0x0000) to 255.9961 ms (0xFFFF)]
			MAX	[0 ms (0x0000) to 255.9961 ms (0xFFFF)]
			RES	[0 ms (0x0000) to 255.9961 ms (0xFFFF)]
			VLIST	[0 ms (0x0000) to 255.9961 ms (0xFFFF)]+
Field	**Value**			
CS	DR_ATTACK_TIME_CONTROL			
OCN:ICN:IPN	OCN = ICN = The Pin channel number on which the AudioControl resides (including the Primary channel 0); IPN = 1			
Parameter Block	Layout 2 (See Section 5.3.2.6.2, “Layout 2 DataPart”)			

5.3.3.7.4.6 RELEASE TIME CONTROL

The Release Time Control is used to determine the recovery response of the DRC on an outgoing channel to a step decrease in the signal level on the corresponding input channel.

Table 5-62: Release Time Control Characteristics

Release Time	SL	RW	OPT	
CUR	M	RoW		
NEXT	O	mRW		
RANGE	M	RO	MIN	[0 ms (0x0000) to 255.9961 ms (0xFFFF)]
			MAX	[0 ms (0x0000) to 255.9961 ms (0xFFFF)]
			RES	[0 ms (0x0000) to 255.9961 ms (0xFFFF)]
			VLIST	[0 ms (0x0000) to 255.9961 ms (0xFFFF)]+
Field	**Value**			
CS	DR_RELEASE_TIME_CONTROL			
OCN:ICN:IPN	OCN = ICN = The Pin channel number on which the AudioControl resides (including the Primary channel 0); IPN = 1			
Parameter Block	Layout 2 (See Section 5.3.2.6.2, “Layout 2 DataPart”)			

5.3.3.7.4.7 LATENCY CONTROL

See Section 5.3.3.1.4, “Latency Control” for a detailed description. XX shall be replaced by DR.

5.3.3.8 PROCESSING UNIT COMMAND

This Command is used to manipulate the AudioControls inside the Processing Unit. The exact layout of the Command is defined in Section 5.3.2.5, “AudioControl Command DataPart Layout.”

The following paragraphs present a detailed description of all possible AudioControls the Processing Unit may incorporate. For each AudioControl, the supported Attributes and their value ranges are specified. Also, the appropriate AudioControl Selector value and the layout type of the Parameter Blocks are listed. The Control Selector codes are defined in Appendix A.23.7, “Processing Unit Control Selectors.”

5.3.3.8.1 UP/DOWN-MIX PROCESSING UNIT

5.3.3.8.1.1 BYPASS CONTROL

See Section 5.3.3.1.1, “Bypass Control” for a detailed description. XX shall be replaced by UD.

5.3.3.8.1.2 CLUSTER CONTROL

The Cluster Control is used to indirectly change the behavior of the Up/Down-mix Processing Unit. The operational mode of the Up/Down-mix Processing Unit is changed so that the outgoing Cluster matches the configuration as indicated by the Cluster Control. See Section 5.3.3.1.2, “Cluster Control” for a detailed description. XX shall be replaced by UD.

5.3.3.8.1.3 CLUSTER ACTIVE CONTROL

This AudioControl shall always be present. See Section 5.3.3.1.3, “Cluster Active Control” for a detailed description. XX shall be replaced by UD.

5.3.3.8.1.4 LATENCY CONTROL

See Section 5.3.3.1.4, “Latency Control” for a detailed description. XX shall be replaced by UD.

5.3.3.8.2 CHANNEL REMAP PROCESSING UNIT

5.3.3.8.2.1 BYPASS CONTROL

See Section 5.3.3.1.1, “Bypass Control” for a detailed description. XX shall be replaced by CR.

5.3.3.8.2.2 CLUSTER CONTROL

The Cluster Control is used to indirectly change the behavior of the Channel Remap Processing Unit. The remapping mode of the Channel Remap Processing Unit is changed so that the outgoing Cluster matches the configuration as indicated by the Cluster Control. See Section 5.3.3.1.2, “Cluster Control” for a detailed description. XX shall be replaced by CR.

5.3.3.8.2.3 CLUSTER ACTIVE CONTROL

This AudioControl shall always be present. See Section 5.3.3.1.3, “Cluster Active Control” for a detailed description. XX shall be replaced by CR.

5.3.3.8.2.4 LATENCY CONTROL

See Section 5.3.3.1.4, “Latency Control” for a detailed description. XX shall be replaced by CR.

5.3.3.8.3 STEREO EXTENDER PROCESSING UNIT

5.3.3.8.3.1 BYPASS CONTROL

See Section 5.3.3.1.1, “Bypass Control” for a detailed description. XX shall be replaced by ST.

5.3.3.8.3.2 WIDTH CONTROL

The Width Control is used to change the spatial appearance of the stereo image, produced by the Stereo Extender Processing Unit.

Table 5-63: Width Control Characteristics

Width	SL	RW	MAN	
CUR	M	RoW		
NEXT	O	mRW		
RANGE	M	RO	MIN	[0 (0x00) to 255 (0xFF)]
			MAX	[0 (0x00) to 255 (0xFF)]
			RES	[0 (0x00) to 255 (0xFF)]
			VLIST	[0 (0x00) to 255 (0xFF)]+
Field	**Value**			
CS	ST_WIDTH_CONTROL			
OCN:ICN:IPN	0:0:0			
Parameter Block	Layout 1 (See Section 5.3.2.6.1, “Layout 1 DataPart”)			

5.3.3.8.3.3 LATENCY CONTROL

See Section 5.3.3.1.4, “Latency Control” for a detailed description. XX shall be replaced by ST.

5.3.3.8.4 MULTI-FUNCTION PROCESSING UNIT

5.3.3.8.4.1 BYPASS CONTROL

The Bypass Control shall be implemented for a Multi-Function Processing Unit. See Section 5.3.3.1.1, “Bypass Control” for a detailed description. XX shall be replaced by MF.

5.3.3.8.4.2 CLUSTER CONTROL

See Section 5.3.3.1.2, “Cluster Control” for a detailed description. XX shall be replaced by MF.

5.3.3.8.4.3 CLUSTER ACTIVE CONTROL

This AudioControl shall always be present. See Section 5.3.3.1.3, “Cluster Active Control” for a detailed description. XX shall be replaced by MF.

5.3.3.8.4.4 ALGO PRESENT CONTROL

The Algo Present Control is a mandatory Read-Only AudioControl that indicates which of the predefined algorithms are currently supported. The CUR Attribute returns a 32-bit **AlgorithmsPresent** bitmap, indicating which of the predefined algorithms are currently available. Multiple bits may be set simultaneously. The CUR Attribute value may change dynamically and generate an interrupt. The bit allocation in the **AlgorithmsPresent** bitmap is as follows:

- D0: Beam Forming.
- D1: Acoustic Echo Cancellation.
- D2: Active Noise Cancellation.
- D3: Blind Source Separation.
- D4: Noise Suppression/Reduction.
- D23..5: Reserved.
- D31..24: Vendor-defined Algorithms.

Table 5-64: Algo Present Control Characteristics

Algo Present	SL	RW	MAN	
CUR	M	RO		
NEXT	P	N/A		
RANGE	I	N/A	MIN	N/A
			MAX	N/A
			RES	N/A
			VLIST	N/A
Field	**Value**			
CS	MF_ALGO_PRESENT_CONTROL			
OCN:ICN:IPN	0:0:0			
Parameter Block	Layout 3 (See Section 5.3.2.6.3, “Layout 3 DataPart”)			

5.3.3.8.4.5 ALGO ENABLE CONTROL

The Algo Enable Control provides a means to selectively enable or disable the implemented algorithms. The CUR Attribute is a 32-bit **EnableAlgorithms** bitmap, where each bit set to one enables the corresponding algorithm. The bit allocations are identical to those of the **AlgorthmsPresent** bitmap as defined in Section 5.3.3.8.4.4, “Algo Present Control” above.

Table 5-65: Enable Algo Control Characteristics

Enable Algo	SL	RW	OPT	
CUR	M	mRW	32-bit **EnableAlgorithms** bitmap	
NEXT	O	mRW		
RANGE	I	N/A	MIN	N/A
			MAX	N/A
			RES	N/A
			VLIST	N/A
Field	**Value**			
CS	MF_ALGO_ENABLE_CONTROL			
OCN:ICN:IPN	0:0:0			
Parameter Block	Layout 3 (See Section 5.3.2.6.3, “Layout 3 DataPart”)			

5.3.3.8.4.6 LATENCY CONTROL

See Section 5.3.3.1.4, “Latency Control” for a detailed description. XX shall be replaced by MF.

5.3.3.9 EXTENSION UNIT COMMAND

Because this specification has no knowledge about the inner workings of the Extension Unit, it is impossible to define a Command that can manipulate specific Extension Unit AudioControls. However, this specification defines several AudioControls the Extension Unit shall or may support.

This Command is used to manipulate the AudioControls inside the Extension Unit. The exact layout of the Command is defined in Section 5.3.2.5, “AudioControl Command DataPart Layout.”

The following paragraphs present a detailed description of all possible AudioControls the Extension Unit may incorporate. For each AudioControl, the supported Attributes and their value ranges are specified. Also, the appropriate Control Selector value and the layout type of the Parameter Blocks are listed. Issuing non-supported Control Selectors to the Extension Unit leads to a Request Error on the Set request part of the Command. The Control Selector codes are defined in Appendix A.23.8, “Extension Unit Control Selectors.”

5.3.3.9.1 BYPASS CONTROL

The Bypass Control shall be implemented for an Extension Unit. See Section 5.3.3.1.1, “Bypass Control” for a detailed description. XX shall be replaced by XU.

5.3.3.9.2 CLUSTER CONTROL

If implemented, the Extension Unit shall always implement the Cluster Control as Read (r). Vendor defined events within the Extension Unit may cause the Cluster Control to change its value and generate an interrupt, informing the Audio Class driver of the change. See Section 5.3.3.1.2, “Cluster Control” for a detailed description. XX shall be replaced by XU.

5.3.3.9.3 CLUSTER ACTIVE CONTROL

This Control shall always be present. See Section 5.3.3.1.3, “Cluster Active Control” for a detailed description. XX shall be replaced by XU.

5.3.3.9.4 LATENCY CONTROL

See Section 5.3.3.1.4, “Latency Control” for a detailed description. XX shall be replaced by XU.

5.3.3.10 CLOCK SOURCE COMMAND

This Command is used to manipulate the AudioControls inside a Clock Source Entity of the Audio Function. The exact layout of the Command is defined in Section 5.3.2.5, “AudioControl Command DataPart Layout.”

The following paragraphs present a detailed description of all possible AudioControls a Clock Source Entity may incorporate. For each AudioControl, the supported Attributes and their value ranges are specified. Also, the appropriate Control Selector value and the layout type of the Parameter Blocks are listed. The Control Selector codes are defined in Appendix A.23.9, “Clock Source Control Selectors.”

5.3.3.10.1 SAMPLING FREQUENCY CONTROL

The Sampling Frequency Control is used to manipulate the actual sampling frequency of the clock signal that is generated by the Clock Source Entity.

For predictable results, this AudioControl should not be written while a related audiostream is active.

Table 5-66: Sampling Frequency Control Characteristics

Sampling Frequency	SL	RW	MAN	
CUR	M	RoW*		
NEXT	O/P*	mRW		
RANGE	M	RO	MIN	[0 Hz (0x00000000) to 4,294,967,295 Hz (0xFFFFFFFF)]
			MAX	[0 Hz (0x00000000) to 4,294,967,295 Hz (0xFFFFFFFF)]
			RES	[0 Hz (0x00000000) to 4,294,967,295 Hz (0xFFFFFFFF)]
			VLIST	[0 Hz (0x00000000) to 4,294,967,295 Hz (0xFFFFFFFF)]+
Field	**Value**			
CS	CS_SAM_FREQ_CONTROL			
OCN:ICN:IPN	0:0:0			
Parameter Block	Layout 3 (See Section 5.3.2.6.3, “Layout 3 DataPart”)			

* In many cases, the Clock Source Entity represents a crystal oscillator-based generator with a single fixed frequency. In some other cases, the Clock Source entity represents an external clock which cannot be controlled by the Audio Function hardware. In all those cases, the CUR Attribute shall be implemented as Read and the NEXT Attribute shall not be supported. However, the RANGE Attribute shall always be supported to indicate the possible values the CUR Attribute may assume.

5.3.3.10.2 CLOCK VALID CONTROL

The Clock Valid Control is used to indicate if the clock signal that is generated by the Clock Source Entity is valid (stable and reliable).

Table 5-67: Status Control Characteristics

Clock Valid	SL	RW	MAN	
CUR	M	RO	[0 (0x00) to 1 (0x01)]; Invalid Clock or Valid Clock respectively.	
NEXT	P	N/A		
RANGE	I	N/A	MIN	N/A
			MAX	N/A
			RES	N/A
			VLIST	N/A
Field	**Value**			
CS	CS_CLOCK_VALID_CONTROL			
OCN:ICN:IPN	0:0:0			
Parameter Block	Layout 1 (See Section 5.3.2.6.1, “Layout 1 DataPart”)			

5.3.3.11 CLOCK SELECTOR COMMAND

This Command is used to manipulate the AudioControls inside a Clock Selector Entity of the Audio Function. The exact layout of the Command is defined in Section 5.3.2.5, “AudioControl Command DataPart Layout.”

The following paragraphs present a detailed description of all possible AudioControls a Clock Selector Entity may incorporate. For each AudioControl, the supported Attributes and their value ranges are specified. Also, the appropriate Control Selector value and the layout type of the Parameter Blocks are listed. The Control Selector codes are defined in Appendix A.23.10,”Clock Selector Control Selectors.”

5.3.3.11.1 SELECTOR CONTROL

The Selector Control allows selecting among several clock signals.

The Selector Control provides the capability to select a clock signal from several incoming clocks on the Clock Selector Unit's Input Pins. The range for this AudioControl is from 0 to the number of Input Pins the Clock Selector Entity supports, as reflected in the Entity's **bNrInputPins** Descriptor field.

Table 5-68: Clock Selector Control Characteristics

Clock Selector	SL	RW	MAN	
CUR	M	RoW	[1 to **bNrInputPins**]	
NEXT	O	mRW		
RANGE	N/A	N/A	MIN	N/A
			MAX	N/A
			RES	N/A
			VLIST	N/A
Field	**Value**			
CS	CX_CLOCK_SELECTOR_CONTROL			
OCN:ICN:IPN	0:0:0			
Parameter Block	Layout 1 (See Section 5.3.2.6.1, "Layout 1 DataPart")			

5.3.3.12 CONNECTOR ENTITY COMMAND

This Command is used to manipulate the AudioControls inside a Connector Entity of the Audio Function. The exact layout of the Command is defined in Section 5.3.2.5, "AudioControl Command DataPart Layout."

The following paragraphs present a detailed description of all possible AudioControls a Connector Entity may incorporate. For each AudioControl, the supported Attributes and their value ranges are specified. Also, the appropriate Control Selector value and the layout type of the Parameter Blocks are listed. The Control Selector codes are defined in Appendix A.23.11, "Connector Control Selectors."

5.3.3.12.1 INSERTION DETECT CONTROL

If implemented, the Insertion Detect Control shall be supported as Read-Only. It is used to report whether a (compatible) plug has been inserted into the receptacle of the Connector. A value of zero indicates that nothing has been inserted. A value of one indicates that a plug has been inserted.

Table 5-69: Insertion Detect Control Characteristics

Insertion Detect	SL	RW	OPT	
CUR	M	RO	[0 (0x00) to 1 (0x01)]; Nothing Inserted or Plug Inserted respectively	
NEXT	P	N/A		
RANGE	I	N/A	MIN	N/A
			MAX	N/A
			RES	N/A
			VLIST	N/A
Field	**Value**			
CS	CO_INSERTION_CONTROL			
OCN:ICN:IPN	0:0:0			
Parameter Block	Layout 1 (See Section 5.3.2.6.1, “Layout 1 DataPart”)			

5.3.3.13 POWER DOMAIN COMMAND

This Command is used to manipulate the AudioControls inside a Power Domain of the Audio Function. The exact layout of the Command is defined in Section 5.3.2.5, “AudioControl Command DataPart Layout.”

The following paragraphs present a detailed description of all possible AudioControls a Power Domain may incorporate. For each AudioControl, the supported Attributes and their value ranges are specified. Also, the appropriate Control Selector value and the layout type of the Parameter Blocks are listed. The Control Selector codes are defined in Appendix A.23.12, “Power Domain Control Selectors.”

5.3.3.13.1 POWER STATE CONTROL

This AudioControl shall be supported when the Audio Function advertises the existence of a Power Domain through the presence of a Power Domain Entity Descriptor. (It is therefore not necessary to advertise the existence of the AudioControl via the regular **bmOptControls** mechanism.)

The Power State Control is used to selectively bring parts of the Audio Function (a Power Domain) into different Power States and to report their current Power State.

Table 5-70: Power State Control Characteristics

Power State	SL	RW	MAN	
CUR	M	mRW	[0 (0x00) to 4 (0x04)]	
NEXT	O	mRW		
RANGE	I	N/A	MIN	N/A
			MAX	N/A
			RES	N/A
			VLIST	N/A
Field	**Value**			
CS	PD _POWER_STATE_CONTROL			
OCN:ICN:IPN	0:0:0			
Parameter Block	Layout 1 (See Section 5.3.2.6.1, “Layout 1 DataPart”)			

5.3.4 AUDIOSTREAMING COMMANDS

The following sections describe the Commands an Audio Function may support for its AudioStreaming Interfaces. The same layout of the Parameter Blocks is used for both the Set and Pull Commands.

AudioStreaming Commands may be directed either to the AudioStreaming Interface or to the associated isochronous data Endpoint, depending on the location of the AudioControl to be manipulated.

5.3.4.1 INTERFACE AUDIOCONTROL COMMAND

This Command is used to manipulate the AudioControls inside an AudioStreaming Interface of the Audio Function. The exact layout of the Command is defined in Section 5.3.2.5, “AudioControl Command DataPart Layout.”

The following paragraphs present a detailed description of all possible AudioControls an AudioStreaming Interface may incorporate. For each AudioControl, the supported Attributes and their value ranges are specified. Also, the appropriate Control Selector value and the layout type of the Parameter Blocks are listed. The Control Selector codes are defined in Appendix A.23.13, “AudioStreaming Interface Control Selectors.”

5.3.4.1.1 ACTIVE ALTERNATE SETTING CONTROL

This AudioControl is used to inform Host software which Alternate Setting of an AudioStreaming Interface is currently active. The main purpose of this AudioControl is to notify the Host (through an interrupt) that the last selected Alternate Setting is no longer valid. There may be a variety of reasons why this could happen. For instance, increasing the sampling frequency of a Clock Source Entity may render the current Alternate Setting of an interface connected to that Clock Source invalid because more bandwidth is now needed than is available in the current Alternate Setting.

This specification does not allow an interface to change from one active Alternate Setting to another without Host intervention. Whenever an Alternate Setting becomes invalid, the interface is required to switch to (idle) Alternate Setting zero. If this situation may occur in the Audio Function, this AudioControl (and the Valid Alternate Settings Control) shall be present. It always provides the currently active Alternate Setting for the interface. The Host software needs to then take appropriate action to reactivate the interface by switching to a valid Alternate Setting. The value of an Active Alternate Setting Control CUR Attribute shall only be either the last set Alternate Setting or zero.

Table 5-71: Active Alternate Setting Control Characteristics

Active Alternate Setting	SL	RW	OPT	
CUR	M	RO	Current Active Alternate Setting: 0 or lastly set Alternate Setting.	
NEXT	P	N/A		
RANGE	I	N/A	MIN	N/A
			MAX	N/A
			RES	N/A
			VLIST	N/A
Field	**Value**			
CS	AS_ACTIVE_ALT_SETTING_CONTROL			
OCN:ICN:IPN	0:0:0			
Parameter Block	Layout 1 (See Section 5.3.2.6.1, “Layout 1 DataPart”)			

5.3.4.1.2 VALID ALTERNATE SETTINGS CONTROL

This AudioControl is used to inform Host software what the currently possible valid Alternate Settings are for an AudioStreaming Interface. If the Active Alternate Setting Control is present, then this AudioControl shall also be present. It always provides a list of currently valid Alternate Settings for the interface. The value of a Valid Alternate Setting Control CUR Attribute is a bitmap, returned in the **bmValidAltSettings** field, that contains a bit for each possible active Alternate Setting. The **bSize** field indicates the length in bytes of the **bmValidAltSettings** field.

A bit set means that this Alternate Setting is currently valid. A bit cleared means that this Alternate Setting is currently not valid. Bit D0 corresponds to Alternate Setting 0 and shall always be set since it is always a possible valid setting. Bit D1 corresponds to Alternate Setting 1. Bit Dm corresponds to Alternate Setting m. All bits that do not correspond to an existing Alternate Setting shall be set to 0. An attempt to set the interface to an invalid Alternate Setting (through the standard Set Interface Command) shall result in a Request Error.

Table 5-72: Valid Alternate Settings Control Characteristics

Valid Alternate Settings	SL	RW	OPT	
CUR	M	RO		
NEXT	P	N/A		
RANGE	I	N/A	MIN	N/A
			MAX	N/A
			RES	N/A
			VLIST	N/A

Field	Value			
CS	AS_VALID_ALT_SETTINGS_CONTROL			
OCN:ICN:IPN	0:0:0			
Parameter Block	Length	1+n		
Offset	**Field**	**Size**	**Value**	**Description**
0	bSize	1	Number	Indicates the number of bytes in the bmValidAltSettings bitmap: n
1	bmValidAltSettings	n	Bitmap	Each set bit represents a currently valid Alternate Setting for the interface.

5.3.4.2 ENDPOINT COMMAND

There is no endpoint-specific Command defined at this time.

5.3.5 ADDITIONAL COMMANDS

5.3.5.1 CLASS-SPECIFIC STRING DESCRIPTOR COMMAND

To overcome the limitations of the standard Get String Descriptor Request (limited to 255 Device-wide strings), this specification provides a class-specific extension to retrieve a larger set of potentially larger String Descriptors from the Audio Function.

The Pull CS String Descriptor Command is used to retrieve class-specific String Descriptors and shall be supported if the Audio Function contains at least one class-specific String Descriptor. These String Descriptors are uniquely identified within the Audio Function through their **wStrDescriptorID** and **iLangID** values.

5.3.5.1.1 CS STRING DESCRIPTOR COMMAND ADDRESSPART LAYOUT

Table 5-73: CS String Descriptor Command AddressPart Layout

Offset	Field	Size	Value	Description
0	wStrDescriptorID	2	Number	String Descriptor ID of the targeted CS String Descriptor.
2	iLang	2	Number	iLang ID for the String Descriptor.
4	wAttribute	2	Constant	STRING Attribute.
6	Reserved	2	Number	Not used. Shall be set to zero.
8	Reserved	2	Number	Not used. Shall be set to zero.

Offset	Field	Size	Value	Description
10	Reserved	2	Number	Not used. Shall be set to zero.

The **wStrDescriptorID** field specifies the String Descriptor's Descriptor ID. For maximum interoperability between this class-specific method to retrieve String Descriptors from the Audio Function and the standard Get String Descriptor Request, all Audio Function related standard String Descriptors, including the LANGID code array at index 0, that can be retrieved using the standard Get String Descriptor Request shall also be retrievable using this class-specific String Descriptor Command by specifying zero in the high byte and the standard String Descriptor index in the low byte of the **wStrDescriptorID** value. Whether other than Audio Function related standard String Descriptors present in the Device can be retrieved using this new method is implementation dependent. Any newly defined class-specific String Descriptor that uses the **wStrDescriptorID** value as an index, shall use **wStrDescriptorID** values starting from 256 onwards.

The **iLangID** field specifies the iLang ID for the String Descriptor. It contains a zero-based index into the LANGID code array as returned by the Device. A Device shall at most support 126 different languages since the LANGID code array is restricted to 254 bytes and each LANGID code takes up 2 bytes. The range of the **iLangID** is therefore from 0 to 125 maximum.

The values specified in the **wStrDescriptorID** and **iLang** fields shall be appropriate to the recipient. Only existing String Descriptor ID values in the Audio Function may be indexed and only appropriate iLang IDs may be used. If the Command specifies an unknown **wStrDescriptorID** or **iLangID** value, the Pull Command Set Request shall return a Request Error.

The **wAttribute** field shall contain the STRING constant. If the Audio Function does not support class-specific String Descriptors, the Pull Command Set Request shall return a Request Error.

All Reserved fields shall be set to zero. If any of the Reserved fields is not zero, the Pull Command Set Request shall return a Request Error.

5.3.5.1.2 CS STRING DESCRIPTOR COMMAND DATAPART LAYOUT

The length of the Descriptor to return is indicated in the **wLength** field of the Pull CS String Descriptor Command Get Request. If the Descriptor is longer than what is indicated in the **wLength** field, only the initial bytes of the Descriptor are returned. If the Descriptor is shorter than what is indicated in the **wLength** field, the Device indicates the end of the control transfer by sending a short packet when further data is requested.

The layout of the Parameter Block follows the class-specific String Descriptor definition as outlined in Section 4.9, "Class-specific String Descriptors."

5.3.5.2 EXTENDED DESCRIPTOR COMMAND

To overcome the limitations of the standard Get Descriptor Command (limited to maximum 255 bytes long), this specification provides a class-specific method to retrieve larger Descriptors from the Audio Function. Also, since Extended Descriptors can report changes dynamically, they may be used whenever there is a need for the Descriptor to indicate that some of its values have changed (even when its length is less than 256 bytes).

The Pull Extended Descriptor Command is used to retrieve Extended Descriptors and shall be supported whenever the Audio Function uses at least one Extended Descriptor. An Extended Descriptor is uniquely identified by its **wDescriptorID** value and is therefore global to the Audio Function and is always retrieved through the AudioControl Interface.

5.3.5.2.1 EXTENDED DESCRIPTOR COMMAND ADDRESSPART LAYOUT

Table 5-74: Extended Descriptor Command AddressPart Layout

Offset	Field	Size	Value	Description
0	wDescriptorID	2	Number	Extended Descriptor ID of the targeted Extended Descriptor.
2	Reserved	2	Number	Not used. Shall be set to zero.
4	wAttribute	2	Constant	EXTENDED_DESCRIPTOR Attribute.
6	Reserved	2	Number	Not used. Shall be set to zero.
8	Reserved	2	Number	Not used. Shall be set to zero.
10	Reserved	2	Number	Not used. Shall be set to zero.

The **wDescriptorID** field specifies the Extended Descriptor ID. The value specified in the **wDescriptorID** field shall be appropriate to the recipient. Only existing Extended Descriptor ID values in the Audio Function may be used. If the Command specifies an unknown **wDescriptorID**, the Pull Command Set Request shall return a Request Error.

The **wAttribute** field shall contain the EXTENDED_DESCRIPTOR constant. If the Audio Function does not support Extended Descriptors, the Pull Command Set Request shall return a Request Error.

All Reserved fields shall be set to zero. If any of the Reserved fields is not zero, the Pull Command Set Request shall return a Request Error.

5.3.5.2.2 EXTENDED DESCRIPTOR COMMAND DATAPART LAYOUT

The length of the Descriptor to return is indicated in the **wLength** field of the Pull Extended Descriptor Command Get Request. If the Descriptor is longer than what is indicated in the **wLength** field, only the initial bytes of the Descriptor are returned. If the Descriptor is shorter than what is indicated in the **wLength** field, the Device indicates the end of the control transfer by sending a short packet when further data is requested.

The layout of the Parameter Block follows the Extended Descriptor definitions as outlined in Section 4.2, “Class-specific Descriptors.”

5.3.5.3 PAGED EXTENDED DESCRIPTOR COMMAND

To accommodate limited resource Host implementations, this specification supports paged access to the Extended Descriptors. This Command allows retrieval of an Extended Descriptor in 256-byte pages at a time. Each page start at a 256-byte boundary and the page numbering is zero-based. A maximum of one page can be retrieved at a time. Therefore, the maximum allowed value in the **wLength** field is 256.

The Pull Paged Extended Descriptor Command shall be supported whenever the Audio Function uses at least one Extended Descriptor.

An Extended Descriptor is uniquely identified by the **wDescriptorID** value and is therefore global to the Audio Function and is always retrieved through the AudioControl Interface.

5.3.5.3.1 PAGED EXTENDED DESCRIPTOR COMMAND ADDRESSPART LAYOUT

Table 5-75: Paged Extended Descriptor Command AddressPart Layout

Offset	Field	Size	Value	Description
0	wDescriptorID	2	Number	Extended Descriptor ID of the targeted Extended Descriptor.
2	wPageNumber	2	Number	Not used. Shall be set to zero.
4	wAttribute	2	Constant	PAGED_EXTENDED_DESCRIPTOR Attribute.
6	Reserved	2	Number	Not used. Shall be set to zero.
8	Reserved	2	Number	Not used. Shall be set to zero.
10	Reserved	2	Number	Not used. Shall be set to zero.

The **wDescriptorID** field specifies the Extended Descriptor ID.

The **wPageNumber** field specifies the zero-based page number of the page to be retrieved.

The values specified in the **wDescriptorID** and **wPageNumber** fields shall be appropriate to the recipient. Only existing Extended Descriptor ID values and Page numbers in the Audio Function may be used. If the Command specifies an unknown **wDescriptorID** or a non-existent Page Number, the Pull Command Set Request shall return a Request Error.

The **wAttribute** field shall contain the PAGE_EXTENDED_DESCRIPTOR constant. If the Audio Function does not support Extended Descriptors, the Pull Command Set Request shall return a Request Error.

All Reserved fields shall be set to zero. If any of the Reserved fields is not zero, the Pull Command Set Request shall return a Request Error.

5.3.5.3.2 PAGED EXTENDED DESCRIPTOR COMMAND DATAPART LAYOUT

The length of the Descriptor to return is indicated in the **wLength** field of the Pull Paged Extended Descriptor Command Get Request and shall therefore be equal to or less than 256. If the Descriptor is longer than what is indicated in the **wLength** field, only the initial bytes of the Descriptor are returned. If the Descriptor is shorter than what is indicated in the **wLength** field, the Device indicates the end of the control transfer by sending a short packet when further data is requested.

The layout of the Parameter Block follows the Extended Descriptor definitions as outlined in Section 4.2, “Class-specific Descriptors.”

6 INTERRUPTS

Interrupts are used to inform the Host that a change has occurred in the current state of the Audio Function. This specification currently defines four different Interrupt Source Types:

- AudioControl CUR Attribute Change: An AudioControl inside the Audio Function has changed its CUR Attribute value (any AudioControl inside an Entity, AudioControl or AudioStreaming Interface, or any AudioControl related to an audio Endpoint).
- Extended Descriptor Change: An Extended Descriptor has changed one or more of its fields.
- Class-specific String Descriptor Change: A class-specific String Descriptor has changed.

A change of state in the Audio Function is most often the result of an event, either user-initiated or Device-initiated. Insertion or removal of a connector is a typical example of a user-initiated event. As a result, the Host could switch selectors or mixers so as to play audio out of the just inserted device (e.g. a headphone) and stop playing audio out of the current device (e.g. a speaker set).
An example of a Device-initiated event is an external device (e.g. an A/V receiver) automatically switching from PCM to AC-3 encoded data on its optical digital output, depending on the material that is currently being played. If this device is connected to the optical digital input of an Audio Function that has auto-detect capabilities, the interface on that Audio Function may need to be reconfigured (e.g., to start the AC-3 decoding process), causing other interfaces to change some aspect of their format, or even become unusable. The Device could issue an interrupt, letting the Host know that the Audio Function needs reconfiguration.

6.1 INTERRUPT MESSAGE

This specification defines four different Interrupt Message Types, one for each defined Interrupt Source Type:

- AudioControl
- Extended Descriptor
- Class-specific String Descriptor

The Interrupt Message consists of a common Interrupt Message Header, followed by an Interrupt Message Body. The layout of the Interrupt Message Body depends on the **wAttribute** field as detailed below.

6.1.1 INTERRUPT MESSAGE HEADER

The interrupt Message Header is always six bytes in length.

The **wLength** field contains the total length of the Interrupt Message, in bytes.

The **wAttribute** field shall be set to the Attribute value that caused the interrupt.

The **bSourceNumber** field shall contain the AudioControl or AudioStreaming interface number of the interface that contains the source of the interrupt.

The **Body** field the layout is defined below for the different Interrupt Source Types.

Note: The MaxPacketSize of the interrupt endpoint shall be set such that all Interrupt Messages that the Device is able to generate fit within the endpoint buffer.

Table 6-1: Interrupt Message Header Format

Offset	Field	Size	Value	Description
0	wLength	2	Number	Size of this Interrupt Message, in bytes: 6+n.
2	wAttribute	2	Number	The Attribute that caused the interrupt.
4	bSourceNumber	1	Number	The Interface number of the interface from which the interrupt originates.
5	Reserved	1	Number	Not used. Shall be set to zero.
6	Body	n	N/A	Source-specific Interrupt Message Body.

6.1.2 AUDIOCONTROL INTERRUPT MESSAGE BODY

The **wAttribute** field shall be set to CUR when the interrupt is caused by a change in the CUR Attribute of an AudioControl. It shall be set to RANGE when the interrupt is caused by a change in the RANGE Attribute of an AudioControl.

The **wEntityID** field contains the EntityID of the Entity that contains the AudioControl from which the interrupt originates. This is only applicable when the **bSourceNumber** field indicates the AudioControl interface number. The **wEntityID** field shall be set to zero otherwise (not used).

The **wControlSelector** field contains the Control Selector value (CS) of the AudioControl from which the interrupt originates.

The **wOCN** field contains the Output Channel Number of the AudioControl from which the interrupt originates.

The **wICN** field contains the Input Channel Number of the AudioControl from which the interrupt originates.

The **wIPN** field contains the Input Pin Number of the AudioControl from which the interrupt originates.

The **ParamBlock** field is only meaningful if the **wAttribute** field specifies CUR and the AudioControl uses one of the predefined DataPart Layouts 1, 2, or 3 as defined in Section 5.3.2.5, “AudioControl Command DataPart Layout.” In this case, the **ParamBlock** field contains the value of the CUR Attribute of the AudioControl (one, two, or four bytes).

If the AudioControl uses a custom DataPart Layout, then the **ParamBlock** field is not present. The Host should query the AudioControl directly to obtain the AudioControl’s most recent Attribute value(s) and to re-enable interrupt generation.

See also Section 6.2, “Interrupt Behavior” for the timing details for the value(s) returned in the **ParamBlock** field.

Table 6-2: AudioControl Interrupt Message Format

Offset	Field	Size	Value	Description
0	wLength	2	Number	Size of this Interrupt Message, in bytes: 16+n (n = 0|1|2|4).
2	wAttribute	2	Number	The AudioControl Attribute from which the interrupt originates.
4	bSourceNumber	1	Number	The interface number of the interface from which the interrupt originates.
5	Reserved	1	Number	Not used. Shall be set to zero.

Offset	Field	Size	Value	Description
6	wEntityID	2	Number	ID of the Entity that contains the AudioControl from which the interrupt originates.
8	wControlSelector	2	Number	The Control Selector value of the AudioControl from which the interrupt originates.
10	wOCN	2	Number	The OCN of the AudioControl from which the interrupt originates.
12	wICN	2	Number	The ICN of the AudioControl from which the interrupt originates.
14	wIPN	2	Number	The IPN of the AudioControl from which the interrupt originates.
16	DataPart	n	Number	The DataPart for the AudioControl, if present.

6.1.3 EXTENDED DESCRIPTOR INTERRUPT MESSAGE BODY

The **wAttribute** field shall be set to EXTENDED_DESCRIPTOR.

The **wDescriptorID** field contains the Descriptor ID of the Extended Descriptor from which the interrupt originates.

The Host should issue a Pull Extended Descriptor Command to obtain the most recent version of the Descriptor and to re-enable interrupt generation.

Table 6-3: Extended Descriptor Interrupt Message Format

Offset	Field	Size	Value	Description
0	wLength	2	Number	Size of this Interrupt Message, in bytes: 8.
2	wAttribute	2	Constant	EXTENDED_DESCRIPTOR.
4	bSourceNumber	1	Number	The interface number of the interface from which the interrupt originates.
5	Reserved	1	Number	Not used. Shall be set to zero.
6	wDescriptorID	2	Number	ID of the Extended Descriptor from which the interrupt originates.

6.1.4 CLASS-SPECIFIC STRING DESCRIPTOR INTERRUPT MESSAGE BODY

The **wAttribute** field shall be set to CLASS_SPECIFIC_STRING_DESCRIPTOR.

The **wStrDescrID** field contains the Descriptor ID of the Class-specific String Descriptor from which the interrupt originates.

The Host should issue a Get Class-specific String Descriptor Command to obtain the most recent version of the String and to re-enable interrupt generation.

Table 6-4: Class-specific String Descriptor Interrupt Message Format

Offset	Field	Size	Value	Description
0	wLength	2	Number	Size of this Interrupt Message, in bytes: 8.

Offset	Field	Size	Value	Description
2	wAttribute	2	Constant	CLASS_SPECIFIC_STRING_DESCRIPTOR.
4	bSourceNumber	1	Number	The interface number of the AudioControl interface from which the interrupt originates.
5	Reserved	1	Number	Not used. Shall be set to zero.
6	wStrDescrID	2	Number	ID of the Class-specific String Descriptor from which the interrupt originates.

6.2 INTERRUPT BEHAVIOR

If the Host directly changes the CUR Attribute on any AudioControl, that AudioControl shall not generate an interrupt. However, any AudioControl CUR or RANGE Attribute within the Audio Function that changes value as a *side effect* of the Host performing the direct Attribute change shall generate an interrupt.

Extended Descriptor and Class-specific String changes shall never create any side effects within the Audio Function.

An Interrupt Message is generated when an event (external or as a side effect of another Host-initiated change) occurs that changes the value of an Attribute.

For AudioControls that use a predefined DataPart, once the Audio Function has detected a change and therefore started the Interrupt Message generation process, the Audio Function ignores any additional changes to that interrupt source, i.e., it does not create an internal interrupt queue to keep track of these changes. The Audio Function then captures the most current value of the interrupt source (Control CUR DataPart) and simultaneously re-enables the interrupt generation capability of the interrupt source. The Audio Function then creates the AudioControl Interrupt Message with that captured CUR DataPart and posts that Message to the interrupt Endpoint. This strategy guarantees that the value delivered in the Interrupt Message is reflecting all changes to it that may have occurred between the moment the first change was detected (and the interrupt process was initiated) and the moment the Interrupt Message was effectively scheduled to be delivered to the Host.

For AudioControls that use a custom DataPart and for Extended and Class-specific String Descriptor interrupt sources, once the Audio Function has detected a change and therefore started the Interrupt Message generation process, it ignores any additional changes to that interrupt source, i.e., it does not create an internal interrupt queue to keep track of these changes. It then creates the corresponding Interrupt Message and posts that Message to the interrupt Endpoint. The interrupt generation for that interrupt source shall remain disabled until the Host retrieves the interrupt source value from the interrupt source through the appropriate Pull Command.

7 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 two groups according to Type.

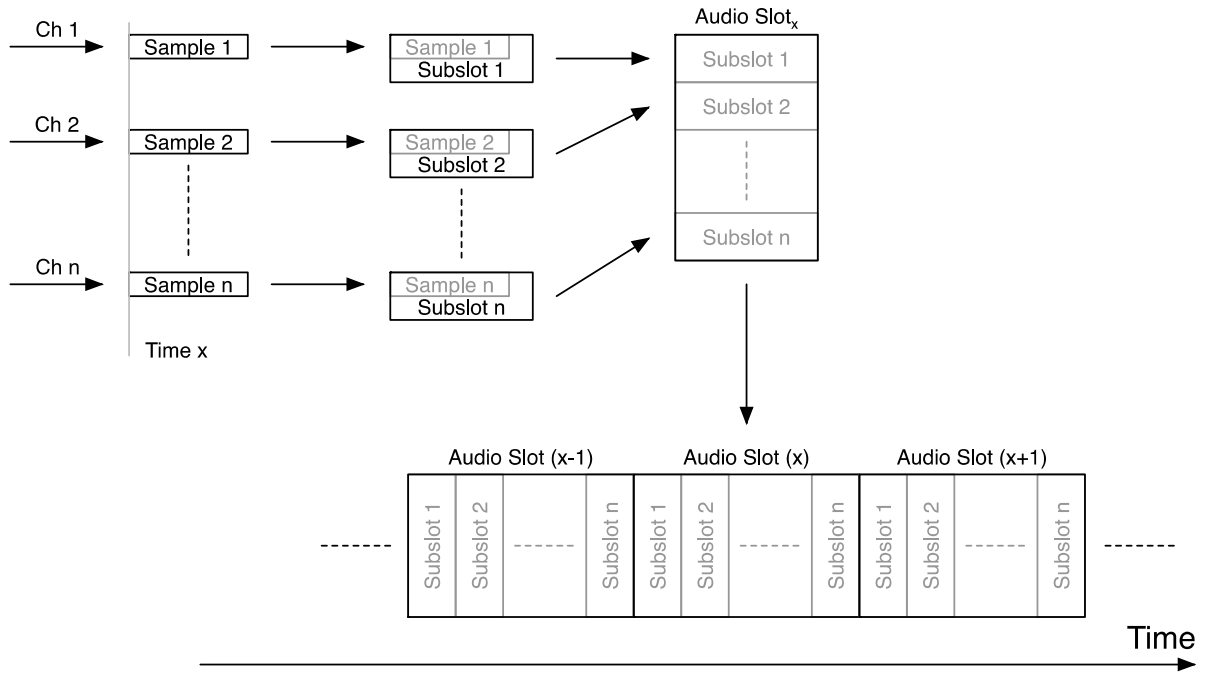
Note: Type II and Type IV Audio Data Formats are no longer supported in this version of the specification.

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 mapping schemes may be used to transform the audio samples into symbols.

Note: Mapping is considered to take place on a per-audio-sample base. Each audio sample generates one symbol (e.g., A-law companding where a 16-bit audio sample is mapped 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 in audio subslots. These audio subslots are then interleaved, according to the cluster channel ordering as described in this 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 7-1: Type I Audio Stream



The second group, Type III, contains Audio Data Formats that use encapsulation as described in the ISO/IEC 61937 standard before being sent over USB. One or more non-PCM encoded audio data streams are packed into "pseudo-stereo samples" and transmitted as if they were real stereo PCM audio samples. The sampling frequency of these pseudo samples (transport sampling frequency, as reported by the Clock Frequency Control of the

associated Clock Source Entity) either matches the sampling frequency of the original non-encoded PCM audio data streams (native sampling frequency) or there is an integer ratio relationship between them. Therefore, clock recovery at the receiving end is relatively easy.

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

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

7.1 SERVICE INTERVAL AND SERVICE INTERVAL PACKET DEFINITIONS

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

A Service Interval is defined as:

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

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

A Service Interval Packet is defined as the amount of isochronous data that is transported during a Service Interval. For high-speed high-bandwidth Endpoints, the Service Interval Packet is the concatenation of the one to three physical packets that are transferred over the bus in a Service Interval.

Note: For high-speed high-bandwidth Endpoints, the Service Interval is equal to the Bus Interval since the **bInterval** field is required to be set to one.

For SuperSpeed and SuperSpeed+ Endpoints, the Service Interval Packet includes all data transferred in the Service Interval, including bursts and multipliers.

7.2 SIMPLE AUDIO DATA FORMATS

7.2.1 TYPE I FORMATS

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

7.2.1.1 USB PACKETS

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

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

7.2.1.2 CONTINUOUS AND BURST MODE

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

- *Continuous Mode*: occurs when the Service Interval is smaller or equal to one USB Frame time of 1 ms. This mode minimizes buffer requirements and latency at the potential expense of higher power consumption. Note however that choosing a Service Interval that is smaller than one USB Frame time may result in excessive system level interrupts.
- *Burst Mode*: occurs when the Service Interval is larger than one USB Frame time. This mode provides for opportunities to save more power by allowing various system components to enter low power states for extended periods of time at the expense of larger buffer sizes and increased latency.

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

In cases where an Explicit Feedback is required to operate the isochronous data endpoint, the Service Interval of the Feedback endpoint shall be equal to or larger than the Service Interval of the corresponding isochronous data endpoint.

7.2.1.2.1 SERVICE INTERVAL PACKET SIZE CALCULATION

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

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

where T_{SI} is the duration of an SI and Δt is the sample time ($1/F_s$). In some 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 two audio slots, that is, $\Delta n_i = 2$. This implies that n_i may vary between $INT(n_{av}) - 1$ (small SIP), $INT(n_{av})$ (medium SIP) and $INT(n_{av}) + 1$ (large SIP). For all i:

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

Furthermore, a large SIP shall be generated as soon as it becomes available. Typically, a source will generate 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.

Note that in some cases (for example, asynchronous Endpoint operating at low frequency), the above formula will result in $n_i = 0$ occasionally, i.e., no payload is available for the Service Interval. In this case, a zero-length packet shall be sent in that Service Interval.

Due to possible different notions of time in the source and the sink (they may 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 always accept a large SIP.

Example:

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

Table 7-1: Packetization

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

7.2.1.3 PITCH CONTROL

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

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

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

7.2.1.4 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 (**wSubslotSize**) to 1, 2, 3, 4, or 8 bytes per audio subslot. An audio sample is represented using a number of bits (**wBitResolution**) less than or equal to the total number of bits available in the audio subslot, i.e., **wBitResolution** ≤ **wSubslotSize***8.

AudioStreaming Endpoints shall be constructed in such a way that a valid transfer may take place as long as the reported audio subslot size (**wSubslotSize**) is respected during transmission. If the reported bits per sample (**wBitResolution**) do not correspond with the number of significant bits used during transfer, the device will either

discard trailing significant bits ([actual_bits_per_sample] > **wBitResolution**) or interpret trailing zeroes as significant bits ([actual_bits_per_sample] < **wBitResolution**).

7.2.1.5 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 this specification. All audio subslots shall have the same audio subslot size.

7.2.1.6 AUDIO STREAMS

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

7.2.1.7 TYPE I SUPPORTED FORMATS

The following paragraphs list all currently supported Type I Audio Data Formats. The allowed values for the **wFormat** field of the class-specific AudioStreaming Self descriptor for the different Type I Audio Data Formats can be found in Appendix B.1, “Audio Data Formats .”

7.2.1.7.1 PCM FORMAT

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

7.2.1.7.2 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 (**wBitResolution** = 8). In this case, data is unsigned fixed-point, left-justified in the audio subslot, MSb first. The range is [0..255].

7.2.1.7.3 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 interpreted as 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 (**wBitResolution** = 32; **wSubslotSize** = 4).

7.2.1.7.4 ALAW FORMAT AND μLAW FORMAT

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

7.2.1.7.5 DSD FORMAT

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

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

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

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

The subslot data is a 64-bit value in little-endian notation where bit D0 (LSb) is the most recent and bit D63 (MSb) is the least recent. This will result in the most recent bit D0 to appear on the USB wire first.

7.2.2 TYPE III FORMATS

These formats are based upon the ISO/IEC 61937 standard. The ISO/IEC 61937 standard describes a method to transfer non-PCM encoded audio bit streams over an ISO/IEC 60958 digital audio interface, together with the transfer of the accompanying “Channel Status” and “User Data.”

The ISO/IEC 60958 standard specifies a widely used method of interconnecting digital audio equipment with two-channel linear PCM audio. The ISO/IEC 61937 standard describes a way in which the ISO/IEC 60958 interface shall be used to convey non-PCM encoded audio bit streams for consumer applications.

The same basic techniques used in ISO/IEC 61937 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.

7.2.2.1 TYPE III SUPPORTED FORMATS

Refer to the ISO/IEC 60958 and ISO/IEC 61937 (several parts) specifications for detailed format information. The allowed values for the **wFormat** field of the class-specific AudioStreaming Self descriptor for the different Type III Audio Data Formats can be found in Appendix B.1, “Audio Data Formats .”

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

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

7.3 EXTENDED AUDIO DATA FORMATS

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

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

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

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

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

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 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 the Extended AudioSlots of the Extended Type I SIP (D2=0b1) or not (D2=0b0). This bit shall be set to zero (D2=0b0) for Extended Type III formats.
- Bits D15..3 are reserved.

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

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

If present, the Header immediately follows the SIPDescriptor. There is only one Header allowed per SIP. The Header may consist of one or more SubHeaders, each having a different purpose. Each SubHeader shall start with a **wSubHeaderID** field, uniquely identifying the purpose of the SubHeader. The length of each SubHeader and the semantics of the remaining fields in the SubHeader are defined by this specification. The structure and composition of the Header and the order in which the SubHeaders appear in the Header may change from SIP to SIP.

Table 7-2: SIPDescriptor Layout

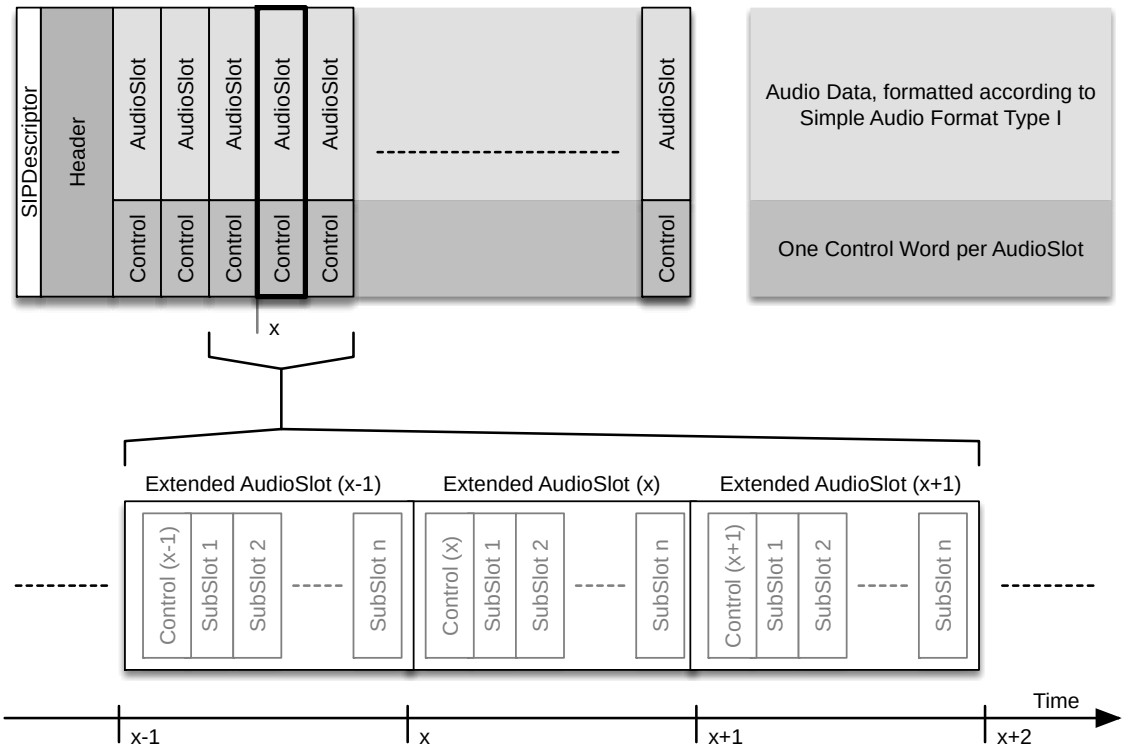
Offset	Field	Size	Value	Description
0	wFlags	2	Bitmap	D0: Header present. D1: AudioSlot present. D2: Control Stream present. D15..3: Reserved.
2	wHeaderLength	2	Number	Total length of the Header, in bytes.

7.3.1 EXTENDED TYPE I FORMATS

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

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

Figure 7-2: Extended Type I Format



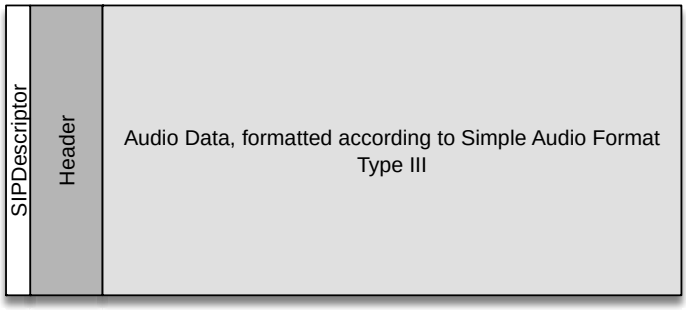
An Extended AudioSlot is the concatenation of a Control Word, followed by the Type I AudioSlot. The Control Stream therefore consists of a sequence of Control Words, where each Control Word is synchronous to its associated AudioSlot. There are as many Control Words per SIP as there are AudioSlots in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

7.3.2 EXTENDED TYPE III FORMATS

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

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

Figure 7-3: Extended Type III Format



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

7.4 CLASS-SPECIFIC AUDIOSTREAMING SELF DESCRIPTOR

The **wStartDelayUnits** field indicates the units that are used for the **wStartDelay** field, immediately following. The **wStartDelay** field indicates how long it takes this Interface to reliably produce valid outgoing data or effectively consume incoming data. This time is measured using as a time reference the first IN or OUT PID that occurs at the start of an audio stream. See Section 3.14.2, “AudioStreaming Interface.”

The **wFormat** field indicates the Audio Data Format that is supported by this Alternate Setting of the AudioStreaming Interface.

A complete list of currently supported Audio Data Formats, their **wFormat** field value, and their usage can be found in Appendix B, “Audio Data Format Codes.”

For Simple and Extended Type I formats, the **wSubslotSize** field can be set to 1, 2, 3, 4, or 8, and the **wBitResolution** field indicates how many bits of the total number of available bits in the audio subslot are effectively used to convey audio information.

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

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

For Extended Type I and Extended Type III formats, this specification defines several Auxiliary Protocols (See Section 7.5, “Auxiliary Protocols.”). The **wAuxProtocols** field contains a bitmap identifying which Auxiliary Protocols this AudioStreaming interface’s Alternate Setting requires. For each Auxiliary Protocol, the AudioStreaming interface may offer up to two Alternate Settings, one in which the Auxiliary Protocol is required and the other in which it is not. For example, an AudioStreaming Interface may offer two Alternate Settings, one indicating the required use of HDCP and the other indicating that it operates without HDCP.

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

Table 7-3: Class-specific AudioStreaming Self Descriptor

Offset	Field	Size	Value	Description
0	wLength	2	Number	Size of this descriptor in bytes: 28.
2	wDescriptorType	2	Constant	EXT_INTERFACE descriptor type.
4	wDescriptorSubtype	2	Constant	AS_SELF descriptor subtype.
6	wDescriptorID	2	Number	Unique ID for this Descriptor.
8	wStrDescriptorID	2	Number	Unique ID for a String Descriptor.
10	dOptControls	4	Bitmap	D0: Active Alternate Setting Control. D1: Valid Alternate Settings Control. D31..2: Reserved.
14	wStartDelayUnits	2	Number	Indicates the units used for the wStartDelay field: 0: Undefined. 1: Milliseconds. 2: Decoded PCM samples. 3..255: Reserved.

Offset	Field	Size	Value	Description
16	wStartDelay	2	Number	Indicates the time it takes this Interface to reliably produce or consume streamed data. Units used depend on the value of the bStartDelayUnits field.
18	wFormat	2	Number	The Audio Data Format used to communicate with this Alternate Setting of the AudioStreaming Interface. See Section 7, “Audio Data Formats” for further details.
20	wSubslotSize	2	Number	The number of bytes occupied by one audio subslot.
22	wBitResolution	2	Number	The number of effectively used bits from the available bits in an audio subslot.
24	wAuxProtocols	2	Bitmap	Bitmap, indicating which Auxiliary Protocols are required.
26	wControlSize	2	Number	Size of the Control Channel Words, in bytes.

7.5 AUXILIARY PROTOCOLS

7.5.1 HDCP PROTOCOL

The HDCP protocol uses the HDCP SubHeader to convey the periodic Link Synchronization information, required by HDCP 2.3. HDCP 2.3 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 7-4: HDCP SubHeader Layout

Offset	Field	Size	Value	Description
0	wLength	2	Number	Size of this SubHeader, in bytes. 18.
2	wSubHeaderID	2	Number	HDCP_ENCRYPTION
4	wOffset	2	Number	Offset to first full 16-byte encrypted data block in the SIP.
6	dStreamCtr	4	Number	The *streamCtr* value as assigned by the HDCP transmitter.
10	qInputCtr	8	Number	The *inputCtr* value associated with the first full 16-byte encrypted data block in the SIP.

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

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

The **wOffset** 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 **wOffset** 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 **dStreamCtr** field contains the *streamCtr* value associated with this 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 (see Appendix B.3, “Audio Format General Constants”) but may occur more frequently. The presence of the HDCP SubHeader indicates to the HDCP receiver that the content is HDCP-encrypted.

7.6 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 value for the **wFormat** field of the class-specific AudioStreaming Self 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.

7.7 ADDING NEW AUXILIARY PROTOCOLS

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

8 BACKWARDS COMPATIBILITY CONSIDERATIONS

ADC 4.0 compliant Devices are not required to include configurations that are compliant with previous versions of the ADC Specification. In other words, backwards compatibility is optional.

To ensure backwards compatibility with a certain revision level of the Audio Device Class-specification, called the Base Revision Level (BRL), a Device that is also capable of operating its Audio Function at Higher Revision Levels (HRL) than the BRL, is called a multi-mode Device and shall do all of the following.

- Provide the BRL Audio Function Descriptor Set during enumeration that is fully compliant with the BRL Audio Device Class-specification.
- Be able to operate its Audio Function in compliance with and as indicated by the BRL Audio Function Descriptor Set.
- Make available to the Host one or more HRL Audio Function Descriptor Sets that each are fully compliant with their respective HRL Audio Device Class-specifications. How this is accomplished is detailed further below.
- Be able to operate its Audio Function in compliance with those HRL Audio Device Class-specifications as indicated by their respective HRL Audio Function Descriptor Sets, once instructed to do so. How this is accomplished is detailed further below.

For this 4.0 version of the Specification, Revision Levels 2.0 and 4.0 shall be the only Base Revision Levels supported and Revision Level 4.0 shall be the only Higher Revision Level supported if the Device supports Base Revision Level 2.0.

Note: It is anticipated that when future version of this Specification (4.1+) become available, Devices may choose to advertise multiple Higher Revision Level modes of operation to ensure the broadest possible level of compatibility. For example, a Device may advertise a 2.0 BRL mode of operation, together with both 4.0 and 5.0 HRL modes of operation.

Both the BRL Audio Function Descriptor Set and the HRL Audio Function Descriptor Sets follow the familiar ADC layout. All Descriptors in both Sets follow the traditional Descriptor layout as described in the *USB Specifications*, i.e., they start with a **bLength** field, followed by a **bDescriptorType** field. Also, see Section 4.3, “Audio Function Descriptor Set” for more details.

To ensure backwards compatibility, the HRL Audio Function Descriptor Sets shall use the identical footprint, layout, and contents for their standard Interface and Endpoint Descriptors (shown in white in the figures in Section 4.3.1, “Audio Function Descriptor Set Layouts”) as is used for the BRL standard Interface Descriptors. The only difference is that the standard Interface Descriptors now indicate the Higher Revision Level in their **bFunctionProtocol** and **bInterfaceProtocol** fields. Indeed, for the Audio Function to be able to switch from BRL operation to a different HRL operating mode, the resources that are allocated once by the USB stack during BRL enumeration need to match the resource requirements of all the HRL operating modes the Audio Function supports. Therefore, the standard Descriptor Set exposed by the BRL Audio Function Descriptor Set and all the exposed HRL Audio Function Descriptor Sets shall be identical, except for the values in the **bFunctionProtocol** and **bInterfaceProtocol** fields.

The class-specific Descriptors (shown in grey in the figures in Section 4.3.1, “Audio Function Descriptor Set Layouts”) can be substantially different between the BRL and HRL Audio Function Descriptor Sets. Nevertheless, it is imperative that the class-specific Descriptors still accurately reflect the actual BRL or HRL operational functionality respectively.

As already mentioned before, the BRL Audio Function Descriptor Set is provided to the Host during enumeration of the Device, while the HRL Interface Descriptor Sets can be retrieved using a method described further below.

Note: The Interface Association Descriptor of the BRL Audio Function Descriptor Set shall not contain a reference to any MIDI Streaming Interfaces. For more information, see the *Universal Serial Bus Device Class Definition for MIDI Devices*.

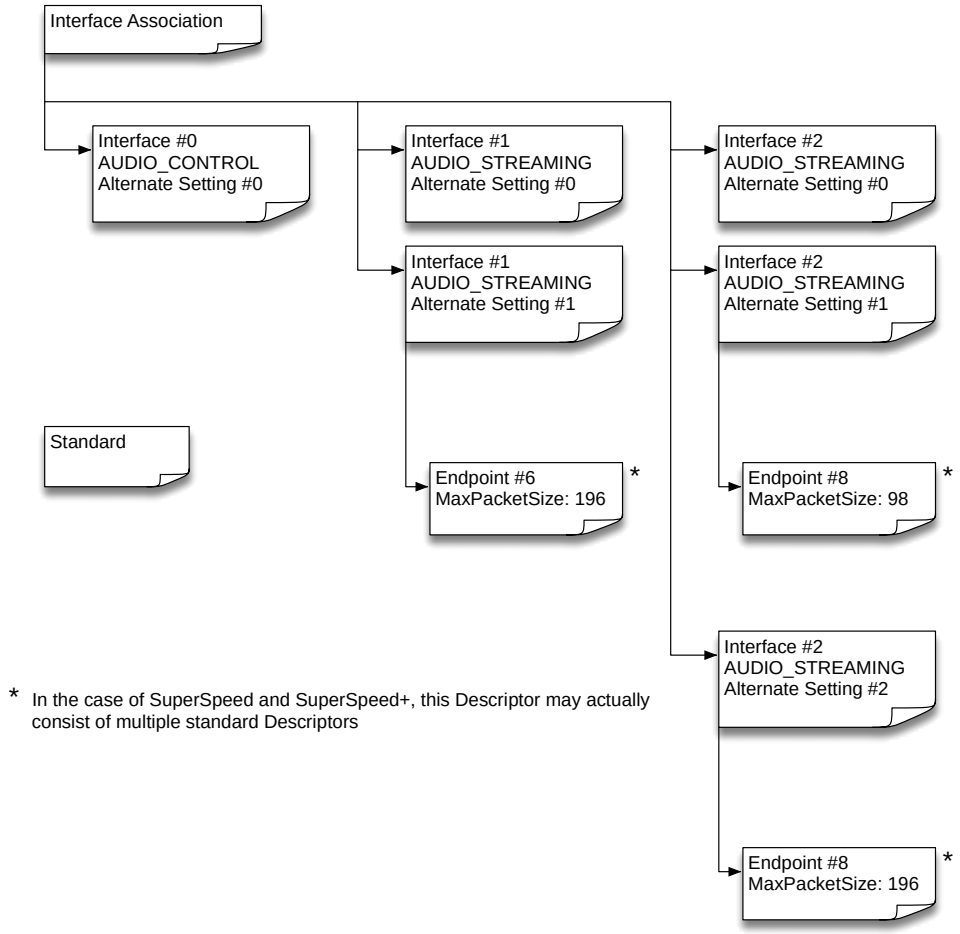
8.1 SIMPLE MULTI-MODE DEVICE EXAMPLE

The following is a hypothetical example to illustrate the procedure outlined above.

Suppose a vendor wants to build a Device that can operate in two modes. A (legacy) mode that is compliant with the ADC 2.0 specification and another mode that is compliant with this ADC 4.0 specification.

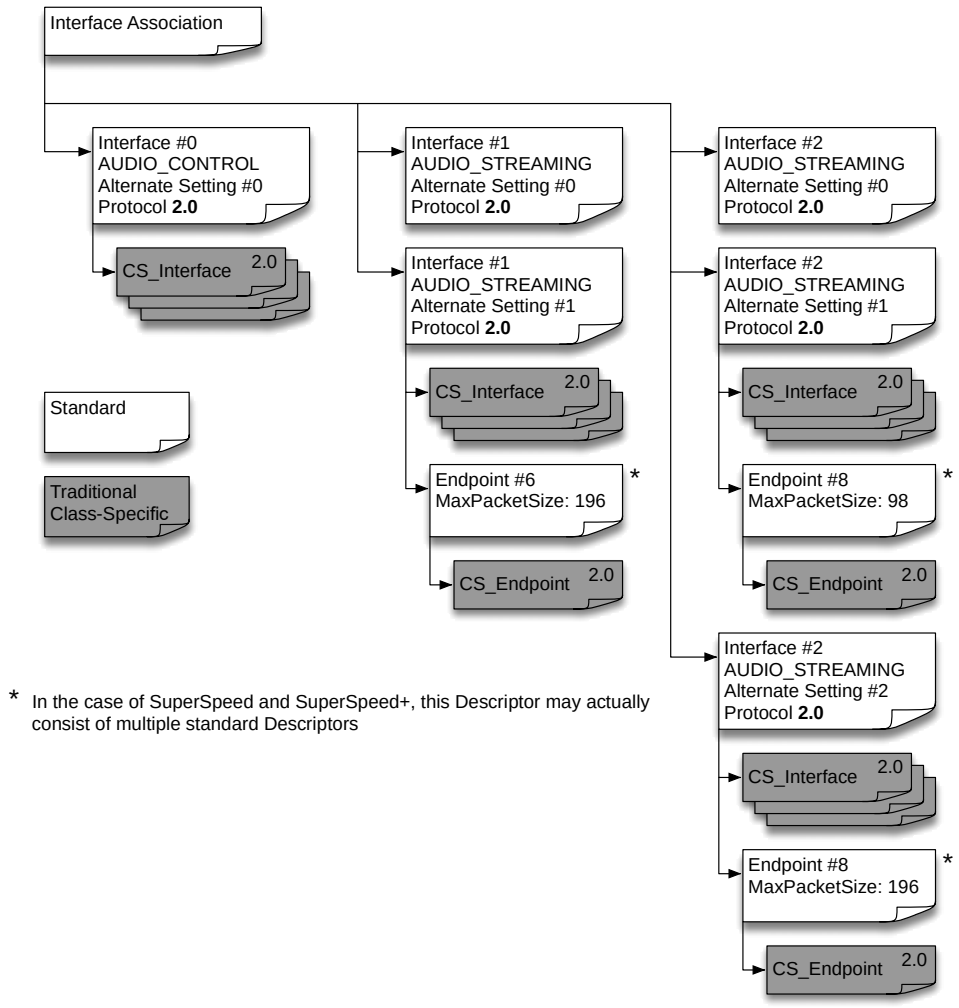
For maximum backwards compatibility, the vendor first creates the set of standard Interface and Endpoint Descriptors as illustrated in Figure 8-1.

Figure 8-1: Standard Interface Descriptor Set Layout



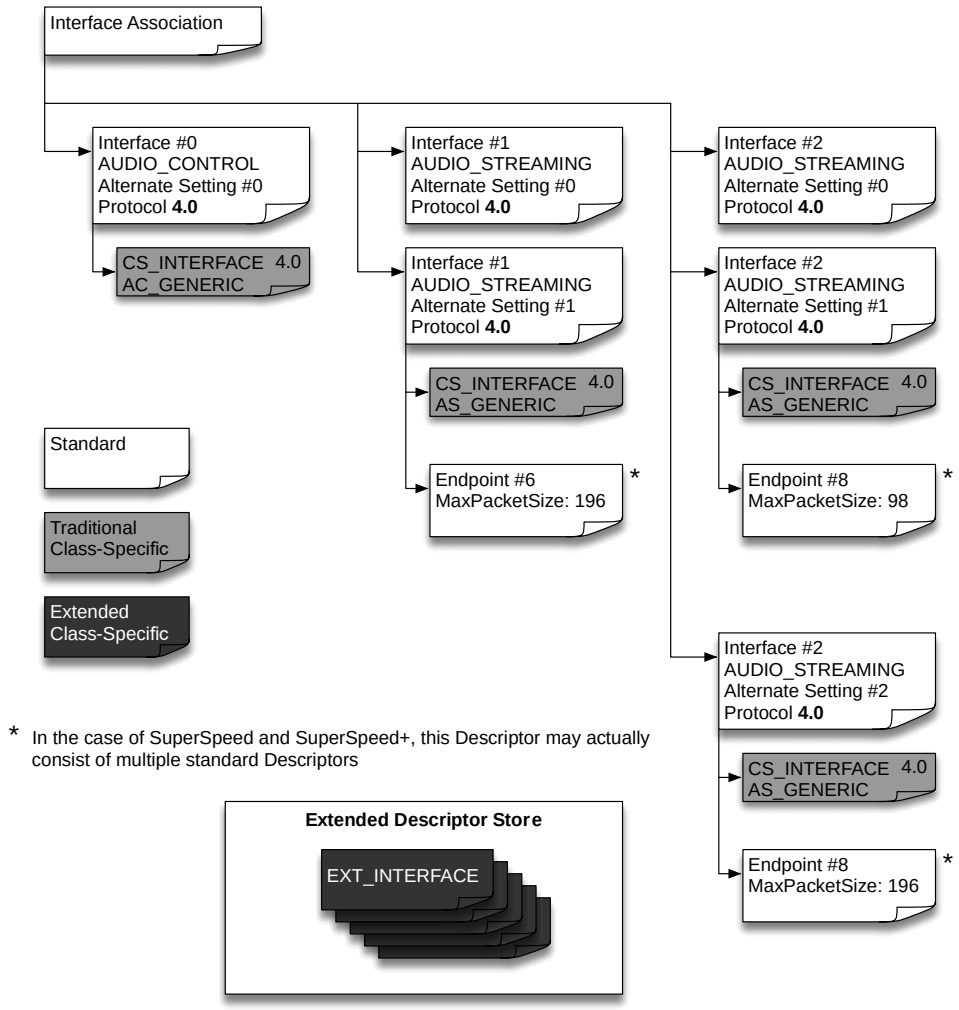
Then for 2.0 BRL operation, the Audio Function exposes the Audio Function Descriptor Set (example) during enumeration as illustrated in Figure 8-2. All the class-specific Interface and Endpoint Descriptors are added to accurately describe the Audio Function in its ADC 2.0 compliant form.

Figure 8-2: ADC 2.0 Audio Function Descriptor Set



For 4.0 HRL operation, the Audio Function exposes the Audio Function Descriptor Set as illustrated in Figure 8-3. All the class-specific Interface and Endpoint Descriptors are added to accurately describe the Audio Function in its ADC 4.0 compliant form.

Figure 8-3: ADC 4.0 Audio Function Descriptor Set



8.2 DISCOVERY OF HIGHER VERSION LEVEL SUPPORT FOR AN AUDIO FUNCTION

If an Audio Function supports a mode at a higher revision level than advertised during enumeration, the Device shall include a new BOS Capability Descriptor, the Higher Revision Level Function Capability Descriptor, as part of its BOS Descriptor. Note that the BOS Descriptor may include multiple HRL Function Capability Descriptors, one for each Higher Revision Level an embedded Audio Function supports, and this for each embedded Audio Function in the Device. The **bConfigurationValue** field, together with the **bFirstInterface** field in the HRL Function Capability Descriptor uniquely identify the Audio Function in the Device to which this HRL Audio Function Capability Descriptor refers.

The BOS Capability Code is assigned by the USB-IF. The assigned code can be found in Appendix A.1, “BOS Capability Codes.”

Note: For this 4.0 revision of the Specification, only one 4.0 HRL Function Capability Descriptor per Audio Function is allowed.

Table 8-1: BOS HRL Function Capability Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	Number	Size of this Descriptor, in bytes: 11.
1	bDescriptorType	1	Constant	DEVICE_CAPABILITY Descriptor type, as defined in the *USB 3 Specification*.
2	bDevCapabilityType	1	Constant	HRL_FUNCTION. Higher Revision Level Audio Function Capability.
3	bConfigurationValue	1	Number	Matches the Configuration Value of exactly one Configuration Descriptor. Only that Configuration will have an Audio Function which may be replaced by this Capability.
4	bFirstInterface	1	Number	Matches the Interface Number of the first USB interface being replaced by this HRL Audio Function.
5	bInterfaceCount	1	Number	The number of USB Interfaces being replaced by this HRL Audio Function.
6	bFunctionClass	1	Constant	AUDIO. Audio Function Class
7	bFunctionSubClass	1	Constant	Audio Function Subclass as defined in the IAD of the HRL Interface Descriptor Set.
8	bFunctionProtocol	1	Number	Audio Function Protocol as defined in the IAD of the HRL Interface Descriptor Set. This indicates the compliance level of the Audio Function being described by this Descriptor.
9	wHRLFuncDescrID	2	Number	The ID of the Extended Descriptor that contains the HRL Audio Function Descriptor Set for the Audio Function.

8.3 RETRIEVAL OF AN HRL AUDIO FUNCTION DESCRIPTOR SET

The HRL Audio Function Descriptor Set consists exclusively of traditional Descriptors, formatted exactly as they would be if they were to be retrieved at enumeration time. The IAD is the first Descriptor in the Set, followed by the standard AudioControl Interface Descriptor and so on.

To make the Set accessible for the Host during normal operation, the HRL Audio Function Descriptor Set is encapsulated within the Extended Function Container Descriptor so that it can be retrieved from the AudioControl Interface using the Get Extended Descriptor Command as defined in Section 5.3.5.2, “Extended Descriptor ”. Note that the HRL Function Container Descriptor can only be retrieved from the Device after the Device is switched into Higher Revision Level mode of operation. Whether the BRL Function Container Descriptor can be retrieved while the device is in Base Revision Level mode of operation is implementation dependent.

The following table outlines the layout of this Function Container Descriptor.

Table 8-2: Function Container Descriptor

Offset	Field	Size	Value	Description
0	wLength	2	Number	Size of this Descriptor, in bytes: 10+n.
2	wDescriptorType	2	Constant	EXT_INTERFACE Descriptor Type.
4	wDescriptorSubType	2	Constant	FUNCTION_CONTAINER Descriptor Subtype.

Offset	Field	Size	Value	Description
6	wDescriptorID	2	Number	Unique ID for this Descriptor.
8	wStrDescriptorID	2	Number	Unique ID for a String Descriptor.
10	HRLAFDescrSet	n	N/A	The HRL Audio Function Descriptor Set.

8.4 SWITCHING THE AUDIO FUNCTION FROM BRL OPERATION TO HRL OPERATION

After inspecting the BOS Descriptor from the Device, the Host can decide to switch the Audio Function into a Higher Revision Level operating mode. This is done through a Set Request operation to the Switch Function Capability that shall be supported whenever the Audio Function supports one or more HRL modes of operation. See Section 5.2.2, “Switch Function” for more details. Note that the switching operation can only be performed once after power up or reset of the Audio Function.

APPENDIX A. AUDIO DEVICE CLASS CODES

This Appendix list all the Audio Device Class constant definitions used by this specification.

A.1 BOS CAPABILITY CODES

Table A-1: BOS Capability Codes

BOS Capability Codes	Value
HRL_FUNCTION	0x12

A.2 AUDIO FUNCTION CLASS CODE

Table A-2: Audio Function Class Code

Audio Function Class Code	Value
AUDIO	0x01

A.3 AUDIO FUNCTION SUBCLASS CODES

Table A-3: Audio Function Subclass Codes

Audio Function Subclass Code	Value
FUNCTION_SUBCLASS_UNDEFINED	0x00
Reserved	0x01..0xFF

A.4 AUDIO FUNCTION PROTOCOL CODES

Table A-4: Audio Function Protocol Codes

Audio Function Protocol Code	Value
FUNCTION_PROTOCOL_UNDEFINED	0x00
AF_VERSION_04_00	IP_VERSION_04_00

A.5 AUDIO INTERFACE CLASS CODE

Table A-5: Audio Interface Class Code

Audio Interface Class Code	Value
AUDIO	0x01

A.6 AUDIO INTERFACE SUBCLASS CODES

Table A-6: Audio Interface Subclass Codes

Audio Interface Subclass Code	Value
INTERFACE_SUBCLASS_UNDEFINED	0x00
AUDIOCONTROL	0x01
AUDIOSTREAMING	0x02
Reserved for MIDI use	0x03

Audio Interface Subclass Code	Value
Reserved	0x04..0xFE

A.7 AUDIO INTERFACE PROTOCOL CODES

Table A-7: Audio Interface Protocol Codes

Audio Interface Protocol Code	Value
IP_VERSION_04_00	0x40
Reserved	All Other Values

A.8 AUDIO CLASS-SPECIFIC TRADITIONAL DESCRIPTOR TYPES

Table A-8: Audio Class-specific Traditional Descriptor Types

Descriptor Type	Value
CS_UNDEFINED	0x20
CS_INTERFACE	0x21
CS_STRING	0x22
Reserved	0x23..0xFF

A.9 AUDIO CLASS-SPECIFIC TRADITIONAL DESCRIPTOR SUBTYPES

Table A-9: Audio Class-specific Traditional Descriptor Subtypes

Descriptor Subtype	Value
SUBTYPE_UNDEFINED	0x00
AC_GENERIC	0x01
AS_GENERIC	0x02

A.10 AUDIO CLASS-SPECIFIC EXTENDED DESCRIPTOR TYPES

Table A-10: Audio Class-specific Extended Descriptor Types

Descriptor Type	Value
EXT_UNDEFINED	0x0000
EXT_INTERFACE	0x0001
EXT_STRING	0x0002
Reserved	0x0003..0xFFFF

A.11 AUDIO CLASS-SPECIFIC EXTENDED DESCRIPTOR SUBTYPES

Table A-11: Audio Class-specific Extended Descriptor Subtypes

Descriptor Subtype	Value
EXT_SUBTYPE_UNDEFINED	0x0000

Descriptor Subtype	Value
AC_SELF	0x0001
INPUT_TERMINAL	0x0002
OUTPUT_TERMINAL	0x0003
TERMINAL_COMPANION	0x0004
MIXER_UNIT	0x0005
SELECTOR_UNIT	0x0006
FEATURE_UNIT	0x0007
EFFECT_UNIT	0x0008
PROCESSING_UNIT	0x0009
EXTENSION_UNIT	0x000A
CLOCK_SOURCE	0x000B
CLOCK_SELECTOR	0x000C
SAMPLE_RATE_CONVERTER	0x000D
CLUSTER	0x000E
CONNECTOR	0x000F
POWER_DOMAIN	0x0010
ENTITY_GROUP	0x0011
COMMIT_GROUP	0x0012
FUNCTION_CONTAINER	0x00FF
AS_SELF	0x0101
AS_VALID_FREQ_RANGE	0x0102
STRING	0x0201
Reserved	All Other Values

A.12 DESCRIPTOR VARIANT TYPES

Table A-12: Descriptor Variant Types

Descriptor Variant Type	Value
VARIANT_NONE	0x0000
VARIANT_INTERFACE	0x0001
VARIANT_ENTITIES	0x0002
Reserved	0x0003..0xFFFF

A.13 CLUSTER DESCRIPTOR SEGMENT TYPES

Table A-13: Cluster Descriptor Segment Types

Segment Type	Value
CLUSTER_SEGMENT_UNDEFINED	0x0000

Segment Type	Value
Reserved for Common Block Segment Types	0x0001..0x00FF
CHANNEL_INFORMATION	0x0101
CHANNEL_AMBISONIC	0x0102
CHANNEL_DESCRIPTION	0x0103
Reserved for Channel Block Segment Types	0x0104..0x01FF
CLUSTER_END_BLOCK	0xFFFF
Reserved	All Other Values

A.14 CHANNEL PURPOSE DEFINITIONS

Table A-14: Channel Purpose Definitions

Channel Purpose	Value
PURPOSE_UNDEFINED	0x0000
GENERIC_AUDIO	0x0001
VOICE	0x0002
SPEECH	0x0003
AMBIENT	0x0004
REFERENCE	0x0005
ULTRASONIC	0x0006
VIBROKINETIC	0x0007
SENSE	0x0008
SILENCE	0xFFFE
NON_AUDIO	0xFFFF
Reserved	All Other Values

A.15 CHANNEL RELATIONSHIP DEFINITIONS

Table A-15: Channel Relationship Definitions

USB Audio Channel Relationship			CEA-861.2 Channel Allocation	
Description	Acronym	Value	Description	Acronym
RELATIONSHIP_UNDEFINED	---	0x0000	---	---
Mono	M	0x0001	---	---
Left	L	0x0002	---	---
Right	R	0x0003	---	---
Array	AR	0x0004	---	---
Headset_Mic	HM	0x0005	---	---
Headset_Mic Left	HML	0x0006	---	---
Headset_Mic Right	HMR	0x0007	---	---

USB Audio Channel Relationship			CEA-861.2 Channel Allocation	
Headset_Mic Center	HMC	0x0008	---	---
Body_Mic	BM	0x0009	---	---
Body_Mic Left	BML	0x000A	---	---
Body_Mic Right	BMR	0x000B	---	---
Body_Mic Center	BMC	0x000C	---	---
Lapel_Mic	LM	0x000D	---	---
Lapel_Mic Left	LML	0x000E	---	---
Lapel_Mic Right	LMR	0x000F	---	---
Lapel_Mic Center	LMC	0x0010	---	---
Pattern_X	PX	0x0011	---	---
Pattern _Y	PY	0x0012	---	---
Pattern _A	PA	0x0013	---	---
Pattern _B	PB	0x0014	---	---
Pattern _M	PM	0x0015	---	---
Pattern _S	PS	0x0016	---	---
Reserved	---	0x0017.. 0x8000	---	---
Front Left	FL	0x8001	Front Left	FL
Front Right	FR	0x8002	Front Right	FR
Front Center	FC	0x8003	Front Center	FC
Front Left of Center	FLC	0x8004	Front Left of Center	FLc
Front Right of Center	FRC	0x8005	Front Right of Center	FRc
Front Wide Left	FWL	0x8006	Front left Wide	FLw
Front Wide Right	FWR	0x8007	Front Right Wide	FRw
Side Left	SL	0x8008	Side Left	SiL
Side Right	SR	0x8009	Side Right	SiR
Surround Array Left	SAL	0x800A	Left Surround	LS
Surround Array Right	SAR	0x800B	Right Surround	RS
Back Left	BL	0x800C	Back Left	BL
Back Right	BR	0x800D	Back Right	BR
Back Center	BC	0x800E	Back Center	BC
Back Left of Center	BLC	0x800F	---	---
Back Right of Center	BRC	0x8010	---	---
Back Wide Left	BWL	0x8011	---	---
Back Wide Right	BWR	0x8012	---	---
Top Center	TC	0x8013	Top Center	TpC

USB Audio Channel Relationship			CEA-861.2 Channel Allocation	
Top Front Left	TFL	0x8014	Top Front Left	TpFL
Top Front Right	TFR	0x8015	Top Front Right	TpFR
Top Front Center	TFC	0x8016	Top Front Center	TpFC
Top Front Left of Center	TFLC	0x8017	---	---
Top Front Right of Center	TFRC	0x8018	---	---
Top Front Wide Left	TFWL	0x8019	---	---
Top Front Wide Right	TFWR	0x801A	---	---
Top Side Left	TSL	0x801B	Top Side Left	TpSiL
Top Side Right	TSR	0x801C	Top Side Right	TpSiR
Top Surround Array Left	TSAL	0x801D	Top Left Surround	TpLS
Top Surround Array Right	TSAR	0x801E	Top Right Surround	TpRS
Top Back Left	TBL	0x801F	Top Back Left	TpBL
Top Back Right	TBR	0x8020	Top Back Right	TpBR
Top Back Center	TBC	0x8021	Top Back Center	TpBC
Top Back Left of Center	TBLC	0x8022	---	---
Top Back Right of Center	TBRC	0x8023	---	---
Top Back Wide Left	TBWL	0x8024	---	---
Top Back Wide Right	TBWR	0x8025	---	---
Bottom Center	BC	0x8026	---	---
Bottom Front Left	BFL	0x8027	Bottom Front Left	BtFL
Bottom Front Right	BFR	0x8028	Bottom Front Right	BtFR
Bottom Front Center	BFC	0x8029	Bottom Front Center	BtFC
Bottom Front Left of Center	BFLC	0x802A	---	---
Bottom Front Right of Center	BFRC	0x802B	---	---
Bottom Front Wide Left	BFWL	0x802C	---	---
Bottom Front Wide Right	BFWR	0x802D	---	---
Bottom Side Left	BSL	0x802E	---	---
Bottom Side Right	BSR	0x802F	---	---
Bottom Surround Array Left	BSAL	0x8030	---	---
Bottom Surround Array Right	BSAR	0x8031	---	---
Bottom Back Left	BBL	0x8032	---	---
Bottom Back Right	BBR	0x8033	---	---
Bottom Back Center	BBC	0x8034	---	---
Bottom Back Left of Center	BBLC	0x8035	---	---
Bottom Back Right of Center	BBRC	0x8036	---	---

USB Audio Channel Relationship			CEA-861.2 Channel Allocation	
Bottom Back Wide Left	BBWL	0x8037	---	---
Bottom Back Wide Right	BBWR	0x8038	---	---
Low Frequency Effects	LFE	0x8039	Low Frequency Effects 1	LFE1
Low Frequency Effects Left	LFEL	0x803A	---	---
Low Frequency Effects Right	LFER	0x803B	Low Frequency Effects 2	LFE2
Headphone Left	HPL	0x803C	---	---
Headphone Right	HPR	0x803D	---	---
Reserved	---	0x803E.. 0xFFFF	---	---

A.16 AMBISONIC COMPONENT ORDERING CONVENTION TYPES

Table A-16: Ambisonic Component Ordering Convention Types

Ambisonic Component Ordering Convention Type	Value
ORD_TYPE_UNDEFINED	0x0000
AMBISONIC_CHANNEL_NUMBER (CAN)	0x0001
FURSE_MALHAM	0x0002
SINGLE_INDEX DESIGNATION (SID)	0x0003
Reserved	0x0004..0xFFFF

A.17 AMBISONIC NORMALIZATION TYPES

Table A-17: Ambisonic Normalization Types

Ambisonic Normalization Type	Value
NORM_TYPE_UNDEFINED	0x0000
max00N	0x0001
SN3D	0x0002
N3D	0x0003
SN2D	0x0004
N2D	0x0005
Reserved	0x0006..0xFFFF

A.18 TERMINAL COMPANION SEGMENT TYPES

Table A-18: Terminal Companion Segment Types

Segment Type	Value
TERMINAL_COMPANION_SEGMENT_UNDEFINED	0x0000
EN_50332-2_SPL_LEVEL	0x0001
EN_50332-2_VOLTAGE_LEVEL	0x0002

Segment Type	Value
CHANNEL_BANDWIDTH	0x0021
CHANNEL_MAGNITUDE_RESPONSE	0x0022
CHANNEL_MAGNITUDE/PHASE_RESPONSE	0x0023
CHANNEL_POSITION_XYZ	0x0024
CHANNEL_POSITION_ R_THETA_PHI	0x0025
TERMINAL_COMPANION_END_BLOCK	0xFFFF
Reserved	All Other Values

A.19 EFFECT UNIT EFFECT TYPES

Table A-19: Effect Unit Effect Types

wEffectType	Value
EFFECT_UNDEFINED	0x0000
PARAM_EQ_SECTION_EFFECT	0x0001
REVERBERATION_EFFECT	0x0002
MOD_DELAY_EFFECT	0x0003
DYN_RANGE_COMP_EXP_EFFECT	0x0004
Reserved	0x0005..0xFFFF

A.20 PROCESSING UNIT PROCESS TYPES

Table A-20: Processing Unit Process Types

wProcessType	Value
PROCESS_UNDEFINED	0x0000
UP/DOWNMIX_PROCESS	0x0001
CHANNEL_REMAP_PROCESS	0x0002
CLUSTER_MODIFICATION_PROCESS	0x0003
STEREO_EXTENDER_PROCESS	0x0004
MULTI_FUNCTION_PROCESS	0x0005
Reserved	0x0006..0xFFFF

A.21 CLASS-SPECIFIC REQUEST CODES

Table A-21: Class-specific Request Codes

Class-specific Request Code	Value
REQUEST_UNDEFINED	0x00
PUSH	0x01
PULL	0x02
COMMIT	0xFE

Class-specific Request Code	Value
SWITCH_FUNCTION	0xFF
Reserved	0x0003..0xFD

A.22 CLASS-SPECIFIC ATTRIBUTE CODES

Table A-22: Class-specific Attribute Codes

Class-specific Attribute Code	Value
ATTRIBUTE_UNDEFINED	0x0000
CUR	0x0001
NEXT	0x0002
RANGE	0x0003
CAP	0x0004
STRING	0x0005
EXTENDED_DESCRIPTOR	0x0006
PAGED_EXTENDED_DESCRIPTOR	0x0007
Reserved	0x0008..0xFFFF

A.23 CONTROL SELECTOR CODES

A.23.1 TERMINAL CONTROL SELECTORS

Table A-23: Terminal Control Selectors

Control Selector	Value
TE_CONTROL_UNDEFINED	0x0000
TE_CLUSTER_CONTROL	0x0001
TE_CLUSTER_ACTIVE_CONTROL	0x0002
TE_VOLTAGE_CONTROL	0x0003
TE_OVERLOAD_CONTROL	0x0004
TE_CLIPPING_CONTROL	0x0005
TE_LATENCY_CONTROL	0x0006
Reserved	0x0007..0xFFFF

A.23.2 MIXER UNIT CONTROL SELECTORS

Table A-24: Mixer Unit Control Selectors

Control Selector	Value
MU_CONTROL_UNDEFINED	0x0000
MU_MIXER_CONTROL	0x0001
MU_CLUSTER_CONTROL	0x0002
MU_CLUSTER_ACTIVE_CONTROL	0x0003

Control Selector	Value
MU_LATENCY_CONTROL	0x0004
Reserved	0x0005..0xFFFF

A.23.3 SELECTOR UNIT CONTROL SELECTORS

Table A-25: Selector Unit Control Selectors

Control Selector	Value
SU_CONTROL_UNDEFINED	0x0000
SU_SELECTOR_CONTROL	0x0001
SU_LATENCY_CONTROL	0x0002
Reserved	0x0003..0xFFFF

A.23.4 FEATURE UNIT CONTROL SELECTORS

Table A-26: Feature Unit Control Selectors

Control Selector	Value
FU_CONTROL_UNDEFINED	0x0000
FU_BYPASS_CONTROL	0x0001
FU_MUTE_CONTROL	0x0002
FU_GAIN_CONTROL	0x0003
FU_BASS_CONTROL	0x0004
FU_MID_CONTROL	0x0005
FU_TREBLE_CONTROL	0x0006
FU_GRAPHIC_EQUALIZER_CONTROL	0x0007
FU_AUTOMATIC_GAIN_CONTROL	0x0008
FU_DELAY_CONTROL	0x0009
FU_BASS_BOOST_CONTROL	0x000A
FU_LOUDNESS_CONTROL	0x000B
FU_INPUT_GAIN_PAD_CONTROL	0x000C
FU_PHASE_INVERTER_CONTROL	0x000D
FU_LATENCY_CONTROL	0x000E
Reserved	0x000F..0xFFFF

A.23.5 SRC UNIT CONTROL SELECTORS

Table A-27: SRC Unit Control Selectors

Control Selector	Value
RU_CONTROL_UNDEFINED	0x0000
RU_LATENCY_CONTROL	0x0001

Control Selector	Value
Reserved	0x0002..0xFFFF

A.23.6 EFFECT UNIT CONTROL SELECTORS

A.23.6.1 PARAMETRIC EQUALIZER SECTION EFFECT UNIT CONTROL SELECTORS

Table A-28: Parametric Equalizer Section Effect Unit Control Selectors

Control Selector	Value
PE_CONTROL_UNDEFINED	0x0000
PE_BYPASS_CONTROL	0x0001
PE_CENTERFREQ_CONTROL	0x0002
PE_QFACTOR_CONTROL	0x0003
PE_GAIN_CONTROL	0x0004
PE_LATENCY_CONTROL	0x0005
Reserved	0x0006..0xFFFF

A.23.6.2 REVERBERATION EFFECT UNIT CONTROL SELECTORS

Table A-29: Reverberation Effect Unit Control Selectors

Control Selector	Value
RV_CONTROL_UNDEFINED	0x0000
RV_BYPASS_CONTROL	0x0001
RV_TYPE_CONTROL	0x0002
RV_LEVEL_CONTROL	0x0003
RV_TIME_CONTROL	0x0004
RV_FEEDBACK_CONTROL	0x0005
RV_PREDELAY_CONTROL	0x0006
RV_DENSITY_CONTROL	0x0007
RV_HIFREQ_ROLLOFF_CONTROL	0x0008
RV_LATENCY_CONTROL	0x0009
Reserved	0x000A..0xFFFF

A.23.6.3 MODULATION DELAY EFFECT UNIT CONTROL SELECTORS

Table A-30: Modulation Delay Effect Unit Control Selectors

Control Selector	Value
MD_CONTROL_UNDEFINED	0x0000
MD_BYPASS_CONTROL	0x0001
MD_BALANCE_CONTROL	0x0002

Control Selector	Value
MD_RATE_CONTROL	0x0003
MD_DEPTH_CONTROL	0x0004
MD_TIME_CONTROL	0x0005
MD_FEEDBACK_CONTROL	0x0006
MD_LATENCY_CONTROL	0x0007
Reserved	0x0008..0xFFFF

A.23.6.4 DYNAMIC RANGE COMPRESSOR/EXPANDER EFFECT UNIT CONTROL SELECTORS

Table A-31: Dynamic Range Compressor/Expander Effect Unit Control Selectors

Control Selector	Value
DR_CONTROL_UNDEFINED	0x0000
DR_BYPASS_CONTROL	0x0001
DR_RATIO_CONTROL	0x0002
DR_MAXAMPL_CONTROL	0x0003
DR_THRESHOLD_CONTROL	0x0004
DR_ATTACK_TIME_CONTROL	0x0005
DR_RELEASE_TIME_CONTROL	0x0006
DR_LATENCY_CONTROL	0x0007
Reserved	0x0008..0xFFFF

A.23.7 PROCESSING UNIT CONTROL SELECTORS

A.23.7.1 UP/DOWN-MIX PROCESSING UNIT CONTROL SELECTORS

Table A-32: Up/Down-mix Processing Unit Control Selectors

Control Selector	Value
UD_CONTROL_UNDEFINED	0x0000
UD_BYPASS_CONTROL	0x0001
UD_CLUSTER_CONTROL	0x0002
UD_CLUSTER_ACTIVE_CONTROL	0x0003
UD_LATENCY_CONTROL	0x0004
Reserved	0x0005..0xFFFF

A.23.7.2 CHANNEL REMAP PROCESSING UNIT CONTROL SELECTORS

Table A-33: Channel Remap Processing Unit Control Selectors

Control Selector	Value
CR_CONTROL_UNDEFINED	0x0000

Control Selector	Value
CR_BYPASS_CONTROL	0x0001
CR_CLUSTER_CONTROL	0x0002
CR_CLUSTER_ACTIVE_CONTROL	0x0003
CR_LATENCY_CONTROL	0x0004
Reserved	0x0005..0xFFFF

A.23.7.3 STEREO EXTENDER PROCESSING UNIT CONTROL SELECTORS

Table A-34: Stereo Extender Processing Unit Control Selectors

Control Selector	Value
ST_CONTROL_UNDEFINED	0x0000
ST_BYPASS_CONTROL	0x0001
ST_WIDTH_CONTROL	0x0002
ST_LATENCY_CONTROL	0x0003
Reserved	0x0004..0xFFFF

A.23.7.4 MULTI-FUNCTION PROCESSING UNIT CONTROL SELECTORS

Table A-35: Multi-Function Processing Unit Control Selectors

Control Selector	Value
MF_CONTROL_UNDEFINED	0x0000
MF_BYPASS_CONTROL	0x0001
MF_CLUSTER_CONTROL	0x0002
MF_CLUSTER_ACTIVE_CONTROL	0x0003
MF_ALGO_PRESENT_CONTROL	0x0004
MF_ALGO_ENABLE_CONTROL	0x0005
MF_LATENCY_CONTROL	0x0006
Reserved	0x0007..0xFFFF

A.23.8 EXTENSION UNIT CONTROL SELECTORS

Table A-36: Extension Unit Control Selectors

Control Selector	Value
XU_CONTROL_UNDEFINED	0x0000
XU_BYPASS_CONTROL	0x0001
XU_CLUSTER_CONTROL	0x0002
XU_CLUSTER_ACTIVE_CONTROL	0x0003
XU_LATENCY_CONTROL	0x0004
Reserved	0x0005..0xFFFF

A.23.9 CLOCK SOURCE CONTROL SELECTORS

Table A-37: Clock Source Control Selectors

Control Selector	Value
CS_CONTROL_UNDEFINED	0x0000
CS_SAM_FREQ_CONTROL	0x0001
CS_CLOCK_VALID_CONTROL	0x0002
Reserved	0x0003..0xFFFF

A.23.10 CLOCK SELECTOR CONTROL SELECTORS

Table A-38: Clock Selector Control Selectors

Control Selector	Value
CX_CONTROL_UNDEFINED	0x0000
CX_SELECTOR_CONTROL	0x0001
Reserved	0x0002..0xFFFF

A.23.11 CONNECTOR CONTROL SELECTORS

Table A-39: Connector Control Selectors

Control Selector	Value
CO_CONTROL_UNDEFINED	0x0000
CO_INSERTION_CONTROL	0x0001
Reserved	0x0002..0xFFFF

A.23.12 POWER DOMAIN CONTROL SELECTORS

Table A-40: Power Domain Control Selectors

Control Selector	Value
PD_CONTROL_UNDEFINED	0x0000
PD_POWER_STATE_CONTROL	0x0001
Reserved	0x0002..0xFFFF

A.23.13 AUDIOSTREAMING INTERFACE CONTROL SELECTORS

Table A-41: AudioStreaming Interface Control Selectors

Control Selector	Value
AS_CONTROL_UNDEFINED	0x0000
AS_ACTIVE_ALT_SETTING_CONTROL	0x0001
AS_VALID_ALT_SETTINGS_CONTROL	0x0002
AS_AUDIO_DATA_FORMAT_CONTROL	0x0003

Control Selector	Value
Reserved	0x0004..0xFFFF

A.24 CONNECTOR TYPES

Table A-42: Connector Types

Connector Type	Value
UNDEFINED	0x0000
2.5 MM PHONE CONNECTOR	0x0001
3.5 MM PHONE CONNECTOR	0x0002
6.35 MM PHONE CONNECTOR	0x0003
XLR/6.35MM COMBO CONNECTOR	0x0004
XLR	0x0005
OPTICAL/3.5MM COMBO CONNECTOR	0x0006
RCA	0x0007
BNC	0x0008
BANANA	0x0009
BINDING POST	0x000A
SPEAKON	0x000B
SPRING CLIP	0x000C
SCREW TYPE	0x000D
DIN	0x000E
MINI DIN	0x000F
EUROBLOCK	0x0010
RJ-11	0x0011
RJ-45	0x0012
TOSLINK	0x0013
HDMI	0x0014
Mini-HDMI	0x0015
Micro-HDMI	0x0016
DP	0x0017
MINI-DP	0x0018
D-SUB	0x0019
THUNDERBOLT	0x001A
LIGHTNING	0x001B
WIRELESS	0x001C
USB LEGACY	0x001D
USB-C	0x001E

Connector Type	Value
SMA	0x001F
SMB	0x0020
LEMO	0x0021
OTHER (CONNECTOR TYPE NOT IN THIS LIST)	0xFFFF
Reserved	All Other Values

A.25 AUDIOCONTROL CAPABILITIES OVERVIEW

Table A-43: AudioControl Capabilities Overview

AudioControl	Support	Comments
AudioControl Interface		
Commit	CR	Required when at least one AudioControl has a NEXT Attribute.
Switch	CR	Required when the Device support at least one HRL operation mode.
Terminal		
Cluster	CR	Required when the Input Terminal represents a non-AudioStreaming Interface.
Cluster Active	MAN	Input Terminal Only
Overload	OPT	
Clipping	OPT	
Latency	MAN*	
Mixer Unit		
Mixer	MAN	At least one Mixer Control shall be present.
Cluster	OPT	
Cluster Active	MAN	
Latency	MAN*	
Selector Unit		
Selector	MAN	
Latency	MAN*	
Feature Unit		
Bypass	OPT	
Mute	OPT	At least one of these AudioControls shall be implemented.
	OPT	
Bass	OPT	
Mid	OPT	
Treble	OPT	
Graphic Equalizer	OPT	

AudioControl	Support	Comments
Automatic Gain	OPT	
Delay	OPT	
Bass Boost	OPT	
Loudness	OPT	
Input Gain Pad	OPT	
Phase Inverter	OPT	
Latency	MAN*	
SRC Unit		
Latency	MAN*	
PEQ Effect Unit		
Bypass	OPT	
Center Frequency	OPT	At least one of these AudioControls shall be implemented.
QFactor	OPT	
Gain	OPT	
Latency	MAN*	
Reverberation Effect Unit		
Bypass	OPT	
Type	OPT	At least one of these AudioControls shall be implemented.
Level	OPT	
Time	OPT	
Delay Feedback	OPT	
Pre-Delay	OPT	
Density	OPT	
Hi_Freq Roll-Off	OPT	
Latency	MAN*	
Modulation Delay Effect Unit		
Bypass	OPT	
Balance	OPT	At least one of these AudioControls shall be implemented.
Rate	OPT	
Depth	OPT	
Time	OPT	
Feedback Level	OPT	
Latency	MAN*	
Dynamic Range Compressor Effect Unit		
Bypass	OPT	

AudioControl	Support	Comments
Compression Ratio	OPT	At least one of these AudioControls shall be implemented.
MaxAmpl	OPT	
Threshold	OPT	
Attack Time	OPT	
Release Time	OPT	
Latency	MAN*	
Up/Down-mix Processing Unit		
Bypass	OPT	
Cluster	OPT	
Cluster Active	MAN	
Latency	MAN*	
Channel Remap Processing Unit		
Bypass	OPT	
Cluster	OPT	
Cluster Active	MAN	
Latency	MAN*	
Stereo Extender Processing Unit		
Bypass	OPT	
Width	MAN	
Latency	MAN*	
Multi-Function Processing Unit		
Bypass	MAN	
Cluster	OPT	
Cluster Active	MAN	
Algo Present	MAN	
Enable Algo	OPT	
Latency	MAN*	
Extension Unit		
Bypass	MAN	
Cluster	OPT	
Cluster Active	MAN	
Latency	MAN*	
Clock Source Entity		
Sampling Frequency	MAN	
Clock Valid	MAN	

AudioControl	Support	Comments
Clock Selector Entity		
Selector	MAN	
Connector Entity		
Insertion	OPT	
Power Domain Entity		
Power State	MAN	
AudioStreaming Interface		
Active Alternate Settings	OPT	
Valid Alternate Settings	OPT	
AudioStreaming Endpoint		
None	-	

* The Latency Control is either supported on all Terminals and Units or it is not supported anywhere.

A.26 INTERRUPT SOURCE TYPES

Table A-44: Interrupt Source Types

Connector Type	Value
SOURCE_TYPE_UNDEFINED	0x0000
AUDIOCONTROL_CUR	0x0001
AUDIOCONTROL_RANGE	0x0002
EXTENDED_DESCRIPTOR	0x0003
CLASS_SPECIFIC_STRING_DESCRIPTOR	0x0004
Reserved	0x0005..0xFFFF

APPENDIX B. AUDIO DATA FORMAT CODES

This Appendix list all the Audio Data Format Code constants used by this specification.

B.1 AUDIO DATA FORMATS

Table B-1: Audio Data Formats in the wFormat Field and Usage

Name	wFormat Value	Type
PCM	0x0000	I
PCM8	0x0001	I
IEEE_FLOAT	0x0002	I
ALAW	0x0003	I
MULAW	0x0004	I
DSD	0x0005	I
Reserved	0x0006..0x00FF	I
PCM_IEC60958	0x0100	III
AC-3	0x0101	III
MPEG-1_Layer1	0x0102	III
MPEG-1_Layer2/3 or MPEG-2_NOEXT	0x0103	III
MPEG-2_EXT	0x0104	III
MPEG-2_AAC_ADTS	0x0105	III
MPEG-2_Layer1_LS	0x0106	III
MPEG-2_Layer2/3_LS	0x0107	III
DTS-I	0x0108	III
DTS-II	0x0109	III
DTS-III	0x010A	III
ATRAC	0x010B	III
ATRAC2/3	0x010C	III
WMA	0x010D	III
E-AC-3	0x010E	III
MAT	0x010F	III
DTS-IV	0x0110	III
MPEG-4_HE_AAC	0x0111	III
MPEG-4_HE_AAC_V2	0x0112	III
MPEG-4_AAC_LC	0x0113	III
DRA	0x0114	III
MPEG-4_HE_AAC_SURROUND	0x0115	III

Name	wFormat Value	Type
MPEG-4_AAC_LC_SURROUND	0x0116	III
MPEG-H_3D_AUDIO	0x0117	III
AC4	0x0118	III
MPEG-4_AAC_ELD	0x0119	III
Reserved	0x011A..0x01FF	III
Reserved	0x0200..0xFFFF	---

B.2 SUBHEADER CODES

Table B-2: SubHeader Codes

SubHeader Code	Value
HEADER_UNDEFINED	0x0000
HDCP_ENCRYPTION	0x0001
Reserved	0x0002..0xFFFF

B.3 AUDIO FORMAT GENERAL CONSTANTS

Table B-3: General Constants

Constant Identifier	Value
HDCP_PACKET_HEADER_TIME	512 (ms)