Universal Serial Bus
Device Class Definition
for
Audio Devices

Release 2.0

May 31, 2006

Scope of This Release

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

Contributors

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

Revision History

Revision	Date	Filename	Author	Description
1.7	Sep. 3, 02	Audio17.doc	USB-IF DWG	Initial version. Based on Audio10.doc. This version will be used to capture the areas where the spec needs adjustments. Areas are indicated by comments.
1.7a	Oct. 24, 02	Audio17a.doc	Geert Knapen	Areas are identified where changes need to be made. Some minor changes already introduced.
1.7b	Oct. 24, 02	Audio17b.doc	Geert Knapen	Intermediate version
1.7c	Dec. 10, 02	Audio17c.doc	Geert Knapen	Discussions from 12-18-2002 f2f meeting captured. Additional comments added.
1.7d	Feb. 3, 03	Audio17d.doc	Geert Knapen	Changes from 1.7c accepted. Additional changes introduced.
1.7e	Feb. 19, 03	Audio17e.doc	Geert Knapen	Introduced physical vs. logical channel cluster
1.7f	Feb. 19, 03	Audio17f.doc	Geert Knapen	Accepted all changes in 1.7e. Fixed some typos.
1.7g	Jun. 2, 03	Audio17g.doc	Geert Knapen	Major overhaul with the introduction of the RANGE attribute.
1.7h	Jun. 3, 03	Audio17h.doc	Geert Knapen	Accepted all changes

Revision	Date	Filename	Author	Description
1.7i	Jul. 10, 03	Audio17i.doc	Geert Knapen	Introduced clock domain, interface association descriptor
1.7j	Jul. 10, 03	Audio17j.doc	Geert Knapen	Accepted all changes
1.7k	Sep. 8, 03	Audio17k.doc	Geert Knapen	Introduced Function Subclass codes, extended interrupt usage, cleaned up clock domains and removed clock domain group concept. Replaced by Clock Source Entity.
1.7l	Sep. 10, 03	Audio17l.doc	Geert Knapen	Accepted all the changes
1.7m	Sep. 15, 03	Audio17m.doc	Geert Knapen	Cleaned up Interrupt description
1.7n	Sep. 30, 03	Audio17n.doc	Geert Knapen	Accepted all changes
1.7o	Sep. 30, 03	Audio17o.doc	Geert Knapen	Major rewrite w.r.t. Controls.
1.7p	Nov. 05, 03	Audio17p.doc	Geert Knapen	Accepted all the changes. Added bit pairs for indicating Control availability
1.7q	Nov. 07, 03	Audio17q.doc	Geert Knapen	Introduced the new concept of controlling sampling frequency
1.7r	Dec. 01, 03	Audio17r.doc	Geert Knapen	Accepted all the changes
1.7s	Dec. 10, 03	Audio17s.doc	Geert Knapen	Changed physical-logical cluster mapping. Added explanation on binding between physical buttons and Audio Controls
1.7t	Feb. 04, 04	Audio17t.doc	Geert Knapen	Accepted all changes
1.7u	Feb. 05, 04	Audio17u.doc	Geert Knapen	Introduced Effect Unit. Regrouped some PUs into the EU concept. Added Parametric EQ as an EU. Accepted all changes
1.7v	Mar. 30, 04	Audio17v.doc	Geert Knapen	Full proof-read. Changed formatting and wording throughout the document
1.7w	Mar. 30, 04	Audio17w.doc	Geert Knapen	Accepted all the changes. Added new Function Categories. Added physical cluster desctriptor to AS interface descriptor.
1.7x	Apr. 13, 04	Audio17x.doc	Geert Knapen	Accepted all the changes. Added new Function Categories. Added support for encoders and decoders.
1.7y	Apr. 28, 04	Audio17y.doc	Geert Knapen	Accepted all the changes.
1.7z	May 15, 04	Audio17z.doc	Geert Knapen	Added some fields to encoder descriptors.
1.8	May 26, 04	Audio18.doc	Geert Knapen	Accepted all changes and promoted to 1.8 level

Revision	Date	Filename	Author	Description
1.8a	Sep. 15, 04	Audio18a.doc	Geert Knapen	Corrected some errors in table offsets etc as indicated by Len Layton (CMedia) Identified the need to address ASR converter Unit
1.8b	Mar. 15, 05	Audio18b.doc	Geert Knapen	Minor editorial changes
1.8c	Aug. 10, 05	Audio18c.doc	Geert Knapen	Minor editorial changes
1.8d	Aug. 16, 05	Audio18d.doc	Geert Knapen	Accepted editorial changes, based on F2F meeting review. Added and accepted an ID field for all encoder and decoder descriptors. This ID must also be passed into the requests that address the encoder or decoder.
1.8e	Aug. 17, 05	Audio18e.doc	Geert Knapen	Redid the encoder sections. Added generic latency support. Added SRC Unit.
1.8f	Aug. 31, 05	Audio18f.doc	Geert Knapen	Fixed some heading levels. Added DTS.
1.8g	Sep. 02, 05	Audio18g.doc	Geert Knapen	Added Encoder and Decoder Error Codes. Accepted all the changes.
1.9RC1	Sep. 02, 05	Audio19RC1.doc	Geert Knapen	Republished unchanged as 1.9RC1.
1.9RC2	Oct. 05, 05	Audio19RC2.doc	Geert Knapen	Made several small editorial changes. Accepted all the changes.
1.9	Oct. 07, 05	Audio19.doc	Geert Knapen	Promoted to 1.9 without change.
2.0RC1	May 19, 06	Audio20RC1.doc	Geert Knapen	Addressed and accepted some minor changes. Declared this document as the Release Candidate for the 2.0 version.
2.0	May 31, 06	Audio20.doc	Geert Knapen	Added new Intellectual Property Disclaimer. Final version.

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

Table of Contents

List of Tables

List of Figures

1 Introduction

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 control the audio environment, such as Volume 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. Handling of MIDI data streams over the USB is directly related to audio and thus covered in this document.

1.2 Purpose

The purpose of this document is to describe the minimum capabilities and characteristics an audio device must support to comply with the USB. This document also provides recommendations for optional features.

1.3 Related Documents

- *Universal Serial Bus Specification*, Revision 2.0 (referred to in this document as the *USB Specification*). In particular, see Chapter 5, “USB Data Flow Model” and Chapter 9, “USB Device Framework.”
- *Universal Serial Bus Device Class Definition for Audio Data Formats* (referred to in this document as *USB Audio Data Formats*).
- *Universal Serial Bus Device Class Definition for Terminal Types* (referred to in this document as *USB Audio Terminal Types*).
- ANSI S1.11-1986 standard.
- MPEG-1 standard ISO/IEC 111172-3 1993.
- MPEG-2 standard ISO/IEC 13818-3 Feb. 20, 1997.
- Digital Audio Compression Standard (AC-3), ATSC A/52A Aug. 20, 2001. (available from **Error! Hyperlink reference not valid.**)
- ANSI/IEEE-754 floating-point standard.
- ISO/IEC 60958 International Standard: *Digital Audio Interface and Annexes.*
- ISO/IEC 61937 standard.

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 Specification*.

Audio Channel Cluster	Group of logical audio channels that carry tightly related synchronous audio information. A stereo audio stream is a typical example of a two-channel audio channel cluster.
Audio Control Attribute	Parameter of an Audio Control. Examples are Current, Minimum, Maximum and Resolution attributes of a Volume Control.
Audio Control	Logical object that is used to manipulate a specific audio property. Examples are Volume Control, Mute Control, etc.
Audio data stream	Transport medium that can carry audio information.

Audio Function	Independent part of a USB device that deals with audio-related functionality.
Audio Interface Collection (AIC)	Grouping of a single AudioControl interface, zero or more AudioStreaming interfaces and zero or more MIDIStreaming interfaces that together constitute a complete interface to an audio function.
AudioControl interface (ACI)	USB interface used to access the Audio Controls inside an audio function.
AudioStreaming interface (ASI)	USB interface used to transport audio streams into or out of the audio function.
Effect Unit (EU)	Provides advanced audio manipulation on the incoming logical audio channels.
Entity	Addressable logical object inside an audio function.
Extension Unit (XU)	Applies an undefined process to a number of logical input channels.
Feature Unit (FU)	Provides basic audio manipulation on the incoming logical audio channels.
FUD	Acronym for Feature Unit Descriptor.
Input Pin	Logical input connection to an Entity. Carries a single audio channel cluster.
Input Terminal (IT)	Receptacle for audio information flowing into the audio function.
ITD	Acronym for Input Terminal Descriptor.
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 Left channel, Right Surround channel, etc.
MIDIStreaming interface (MSI)	USB interface that may be used to transport MIDI data streams into or out of the audio function.
Mixer Unit (MU)	Mixes a number of logical input channels into a number of logical output channels.
MUD	Acronym for Mixer Unit Descriptor.
OTD	Acronym for Output Terminal Descriptor.
Output Pin	Logical output connection to an Entity. Carries a single audio channel cluster.
Output Terminal (OT)	An outlet for audio information flowing out of the audio function.
Processing Unit (PU)	Applies a predefined process to a number of logical input channels.
PUD	Acronym for Processing Unit Descriptor.
Selector Unit (SU)	Selects from a number of input audio channel clusters.
SUD	Acronym for Selector Unit Descriptor.

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 subfunctionality.
XUD	Acronym for Extension Unit Descriptor.

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 PC's 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. With the advent of the High Speed USB, consumers can count on this medium to meet all of their audio needs today and into the future. Many applications from communications, to entertainment, to music recording and playback, can take advantage of audio features of the USB.

In principle, a versatile bus specification like the USB provides many ways to propagate and/or control digital audio. 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 to 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 Specification*. The Audio Device Class definition adheres to this synchronization scheme 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 must be present in each USB audio function. It further explains the use of class-specific requests that allow for full audio function control. A number of predefined data formats are listed and fully documented. Each format defines a standard way of transporting audio over the USB. Provisions have been made so that vendor-specific audio formats and compression schemes can be handled.

Many of the changes introduced in Version 2.0 of the USB Specification for Audio Devices take advantage of the new features provided in the USB 2.0 Specification. With the additional bandwidth made available, high speed USB operation allows the transport of multiple channels of high bit rate audio. This expands the range of solutions provided by USB audio devices but also challenges the way in which they operate. In addition to supporting the additional bandwidth, the specification supports new codec types for consumer audio applications, provides numerous clarifications of the original specification and extensions to support various changes in the core specification. The changes are not generally backwards compatible to 1.0 because that would too severely limit this new class of devices.

2.1 Overview of Key Differences between ADC v1.0 and v2.0

The following list is not an exhaustive list of all changes that have been introduced. For complete information, refer to the full specification. Pay special attention to Sections 1 through 6!

- Complete support for high speed operation - no longer are audio class devices limited to full speed operation.
- The notion of physical and logical Audio channel clusters.
- The number of predefined spatial locations has increased. In addition, a virtual spatial location called Raw Data was introduced.
- Use of the interface association descriptor - The standard Interface Association mechanism is used to describe an Audio Interface Collection. The former class specific mechanism was deprecated.
- Descriptor updates: fixed offsets associated with many descriptors and enlarged three byte fields into four bytes.
- Extensive support for interrupts to inform the host about dynamic changes that occur on the different addressable Entities (Clock Entities, Terminals, Units, interfaces and endpoints) inside the audio function.
- More clarification text on the audio function.

- Audio Control Changes.
  - Control attribute changes.
  - Mixer Unit control request (set/get pairs changed).
  - Many updates in the control descriptions.
- Added support for clock domains, clock description and clock control.
- Added additional Audio Controls inside a Feature Unit (Input, Gain, Input Gain Pad …)
- Added bit pairs in descriptors to indicate presence and programmability of every Control
- Prohibited the use of Alternate Setting switching to change sampling frequencies. Instead, Clock Entities are introduced that can be manipulated (through the AudioControl interface) to select operating sampling frequencies.
- Split off the examples in a separate document.
- Allowed binding between physical buttons on the audio function and the corresponding Audio Control. Prescribed how this is done.
- Added an Effect Unit to group algorithms that work on logical channels separately but require multiple parameters to manipulate the effect (as opposed to basic (single parameter) manipulation, performed in a Feature Unit).
- Introduced Parametric Equalizer Section Effect Unit.
- Rearranged Reverb, Modulation Delay and Dynamic Compressor PUs under the new Effect Unit.
- Added the concept of audio function Category. The Category indicates the primary use of the audio function as envisioned by the manufacturer.
- Added the Sampling Rate Converter Unit.
- Added a means to express Latency of individual building blocks within the audio function.
- Added Encoder support.

3 Functional Characteristics

3.1 Introduction

In many cases, audio functionality does not exist as a standalone device. It is one capability that, together with other functions, constitutes a “composite” device. A perfect example of this is a DVD-ROM player, which can incorporate video, audio, data storage, and transport control. The audio function is thus located at the interface level in the device class hierarchy. It consists of a number of interfaces grouping related pipes that together implement the interface to the audio function.

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 addressed through their audio interfaces. Each audio function must have a single AudioControl interface and can have zero or more AudioStreaming and zero or more MIDIStreaming interfaces. The AudioControl (AC) interface is used to access the Audio Controls of the function whereas the AudioStreaming (AS) interfaces are used to transport audio streams into and out of the function. The MIDIStreaming (MS) interfaces can be used to transport MIDI data streams into and out of the audio function. The collection of the single AudioControl interface and the AudioStreaming and MIDIStreaming interfaces that belong to the same audio function is called the Audio Interface Collection (AIC). A device can have multiple Audio Interface Collections active at the same time. These Collections are used to control multiple independent audio functions located in the same composite device. An Audio Interface Collection is described through the standard USB Interface Association mechanism that expresses interface binding via the Interface Association Descriptor (IAD).

Note: All MIDI-related information is grouped in a separate document, *Universal Serial Bus Device Class Definition for MIDI Devices* that is considered part of this specification. The remainder of this document will therefore not mention MIDIStreaming interfaces and their specifics anymore.

The following figure illustrates the concept:

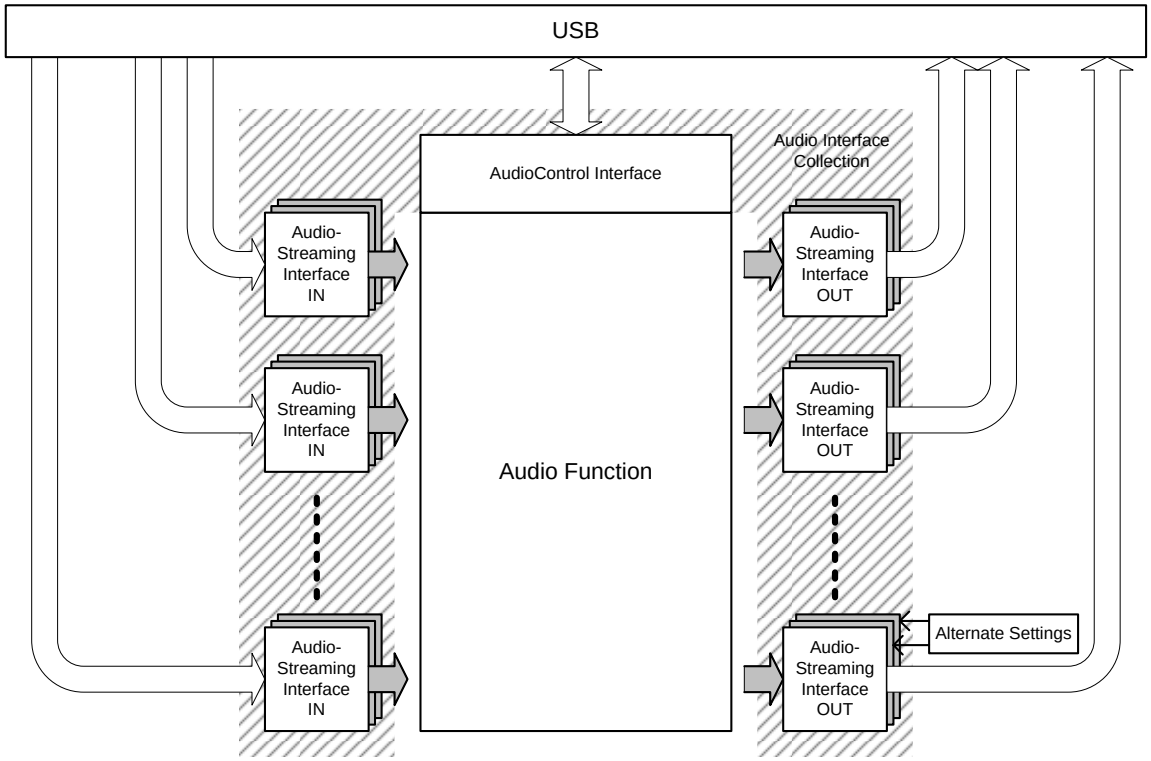

Figure 3-1: Audio Function Global View

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. The AudioStreaming interface is primarily used for transporting audio data between the audio function and the outside world. However, all control data that is related specifically to the streaming behavior is also conveyed through the AudioStreaming interface. In particular, all control data that is used to influence the decoder or encoder process that potentially resides between the actual streaming endpoint and the audio function (e.g. conversion from AC-3 encoded stream to 5.1 physical audio channels) is conveyed through the AudioStreaming interface.

Note that in some cases an AudioStreaming interface is only used to perform controlling functions while no actual data is transported over the interface. A physical S/PDIF connection to the audio function is a typical example. Although the actual audio data is coming in from the outside world (not through the USB), it might be necessary to control some aspects of the S/PDIF connection. In that case, the S/PDIF connection is represented by an AudioStreaming interface so that it becomes addressable through USB.

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.).

3.2 Audio Interface Collection (AIC)

On USB, an audio function is completely defined by its interfaces. An audio function has one AudioControl interface and zero or more AudioStreaming interfaces, grouped into an Audio Interface Collection. The standard USB Interface Association mechanism is used to describe the Audio Interface Collection i.e. to bind those interfaces together. Interface Association is expressed via the standard USB Interface Association Descriptor (IAD). Every Interface Association Descriptor has a FunctionClass, FunctionSubClass and FunctionProtocol field that together identify the function that is represented by the Association. The following paragraphs define these fields for the Audio Device Class.

3.3 Audio Function Class

An Interface Association has a Function Class code assigned to it. This specification requires that the Function Class code be the same as the Audio Interface Class code.

The Audio Function class code is assigned by this specification. For details, see Appendix A.1, “Audio Function Class Code”.

3.4 Audio Function Subclass

The Audio Function class is divided into Function Subclasses. At this moment, the Function SubClass code is not used and must be set to FUNCTION_SUBCLASS_UNDEFINED.

The assigned codes can be found in A.2, “Audio Function Subclass Codes” of this specification. All other Subclass codes are unused and reserved by this specification for future use.

3.5 Audio Function Protocol

The Audio Function class and Subclasses can be further qualified by the Function Protocol code. The Function Protocol code is used to reflect the current version of this specification so that enumeration software can decide which driver versions need to be instantiated.

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

3.6 Audio Interface Class

The Audio Interface class groups all functions that can interact with USB-compliant audio data streams. All functions that convert between analog and digital audio domains can be part of this class. In addition, those functions that transform USB-compliant audio data streams into other USB-compliant audio data streams can be part of this class. Even analog audio functions that are controlled through USB belong to this class.

In fact, for an audio function to be part of this class, the only requirement is that it exposes one AudioControl interface. No further interaction with the function is mandatory, although most functions in the audio interface class will support one or more optional AudioStreaming interfaces for consuming or producing one or more isochronous audio data streams.

The Audio Interface class code is assigned by the USB. For details, see Appendix A.4, “Audio Interface Class Code”.

3.7 Audio Interface Subclass

The Audio Interface class is divided into Subclasses. All audio functions are part of a certain Interface Subclass. The following three Interface Subclasses are currently defined in this specification:

- AudioControl Interface Subclass
- AudioStreaming Interface Subclass
- MIDIStreaming Interface Subclass

The assigned codes can be found in Appendix A.5, “Audio Interface Subclass Codes” of this specification. All other Subclass codes are unused and reserved by this specification for future use.

3.8 Audio Interface Protocol

The Audio Interface class and Subclasses can be further qualified by the Interface Protocol code. The Interface Protocol code is used to reflect the current version of this specification.

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

3.9 Audio Function Category

The Audio Function Category indicates the primary intended use for the audio function. The following Function Categories are currently defined in this specification:

- Desktop Speaker: One or more speakers set up in a small environment to provide audio intended primarily for one person.
- Home Theater: Several speakers set up in a moderately sized environment to provide audio levels significantly louder than a Desktop Speaker setup and intended to be clearly heard by multiple people.
- Microphone: A device set up to record audio from audible sources.
- Headset: A device with at least one speaker and at least one microphone designed to be worn or held by a user to provide personal audio playback and voice input capabilities.
- Telephone: A Headset or handset type device that also connects to a telephone system, (e.g. POTs, PBX, VoIP) capable of making and receiving telephone calls.
- Converter: A device that allows conversion of audio from one electrical or optical format to another electrical or optical format, and/or converting audio data from one encoding format to another (e.g. AC-3 to PCM, etc.).
- Voice/Sound recorder: A device set up with at least one microphone and at least one speaker that is designed to operate, at least some of the time, independently of the Host to record and store audible sources and play back its recorded content.
- IO Box: A device designed to deliver one or more, possibly different, electrical and optical inputs and outputs for connection to other devices.

- Musical Instrument: A musical instrument, e.g. piano, guitar, synthesizer, drum machine, etc.
- Pro-Audio: A device not typically used by consumers of audio, e.g. editing equipment, multitrack recording equipment, etc.
- Audio/Video: The audio from a device that also supplies simultaneous video where the expectation is that the audio is tightly coupled to the video, e.g. a camcorder, a DVD player, a television, etc.
- Control Panel: A device that is used to control the flow of audio through a system of audio devices, such as a mixer panel.
- Other: Any device whose primary purpose is sufficiently different from the above descriptions as to be considered a completely different form of device.

The assigned codes can be found in Appendix A.7, “Audio Function Category Codes” of this specification. All other Category codes are unused and reserved by this specification for future use.

3.10 Clock Domains

A Clock Domain is defined as a zone within which all sampling clocks are derived from the same master clock. Therefore, within the same Clock Domain, all sampling clocks are synchronous and their timing relationship is constant. However, the sampling clocks can be at different sampling frequencies. The master clock can be generated in many different ways. An internal crystal could be the master clock, the USB start of frame (SOF) could be used or even an externally supplied clock could serve as a master clock.

In general, multiple different Clock Domains can 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 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 cannot be synchronized to a start of frame (SOF) or to any other clock in the USB domain.

3.11.2 Synchronous

The clock system of synchronous isochronous audio endpoints can be controlled externally through SOF synchronization. Such an endpoint must lock its sample clock to the 1ms SOF tick. Optionally, a high-speed endpoint could lock its clock to the 125 µs SOF that occurs at the beginning of every microframe to improve accuracy.

3.11.3 Adaptive

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

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 of the audio peripheral devices or functions.

To provide a manageable phase model to the host, an audio function is required to report its internal delay for every AudioStreaming interface. This delay is expressed in number of (micro)frames and is due to the fact that the audio function must buffer at least one (micro)frame worth of samples to effectively remove packet jitter within a (micro)frame. Furthermore, some audio functions will introduce extra delay because they need time to correctly interpret and process the audio data streams (for example, compression and decompression). However, it is required that an audio function introduces only an integer number of (micro)frames of delay. In the case of an audio source function, this implies that the audio function must guarantee that the first sample it fully acquires after SOF*n* (start of (micro)frame *n*) is the first sample of the packet it sends over USB during (micro)frame ($n+\delta$). δ is the audio function's internal delay expressed in (micro)frames. The same rule applies for an audio sink function. The first sample in the packet, received over USB during (micro)frame *n*, must be the first sample that is fully reproduced during (micro)frame ($n+\delta$).

By following these rules, phase jitter is limited to ±1 audio sample. It is up to the host software to synchronize the different audio streams by scheduling the correct packets at the correct moment, taking into account the internal delays of all audio functions involved.

3.13 Audio Function Topology

To be able to manipulate the physical properties of an audio function, its functionality must be divided into addressable Entities. Two types of such generic Entities are identified and are called Units and Terminals. In addition, a special type of Entity is defined. These Entities are called Clock Entities and they are used to describe and manipulate the clock signals inside the audio function.

Units provide the basic building blocks to fully describe most audio functions. Audio functions are built by connecting together several of these Units. A Unit has one or more Input Pins and a single Output Pin, where each Pin represents a cluster of logical audio channels inside the audio function (see Section 3.13.1, "Audio Channel 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, the concept of a Terminal is introduced. There are two types of Terminals. An Input Terminal (IT) is an Entity that represents a starting point for audio channels inside the audio function. An Output Terminal (OT) represents an ending point for audio channels. From the audio function's perspective, a USB endpoint is a typical example of an Input or Output Terminal. It either provides 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. Connection to the Terminal is made through its single Input or Output Pin.

Input Pins of a Unit are numbered starting from one up to the total number of Input Pins on the Unit. The Output Pin number is always one. 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 information, traveling over I/O Pins is 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 together is a guarantee (by construction) that the protocol and format, used over these connections (analog or digital), is compatible on both ends.

Every Unit in the audio function is fully described by its associated Unit descriptor (UD). The Unit descriptor contains all necessary fields to identify and describe the Unit. Likewise, there is a Terminal descriptor (TD) 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 eight different types of standard Units and Terminals that are considered adequate to represent most audio functions available today and in the near future:

- Input Terminal (IT)
- Output Terminal (OT)
- Mixer Unit (MU)
- Selector Unit (SU)
- Feature Unit (FU)
- Sampling Rate Converter Unit
- 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 (CS)
- Clock Selector (CX)
- Clock Multiplier (CM)

A Clock Source provides a certain sampling clock frequency to all or part of the audio function. A Clock Source can represent an internal sampling frequency generator, but it can also represent an external sampling clock signal input to the audio function.

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

A Clock Selector is used to select between multiple sampling clock signals that might 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.

A Clock Multiplier is used to derive a new clock signal with a different frequency from the clock signal at its single Clock Input Pin. It does this by multiplying that clock signal frequency by a numerator P and then dividing it by a denominator Q. The values P and Q are fixed for a given Clock Multiplier. The new clock signal is guaranteed to be synchronous with the input clock signal. A Clock Multiplier has one Input Pin and one Output Pin and their numbers are always one.

By using a combination of Clock Source, Clock Selector, and Clock Multiplier Entities, the most 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 is 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.

Each 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.

Each Clock Entity is described by a Clock Entity descriptor (CED). The Clock Entity descriptor contains all necessary fields to identify and describe the Clock Entity.

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

The ensemble of Unit descriptors, Terminal descriptors and Clock 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, a generic audio driver should be able to fully control the audio function, except for the functionality represented by Extension Units. Those require vendor-specific extensions to the audio class driver.

Important Note:

> The complete set of audio function descriptors provides only a static initial description of the audio function. During operation, a number of events can happen that force the audio function to change its state. Host software must 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 any and all state changes to Host software.

Figure 3-2, "Inside the Audio Function" illustrates the concepts defined above. Using the iconic symbols defined further, it describes a hypothetical audio function that incorporates 16 Entities: three Input Terminals, five Units, three Output Terminals, two Clock Sources, a Clock Selector, and two Clock Multipliers. Each Entity has its unique ID (from 1 to 16) and descriptor that fully describes the functionality of the Entity and also how that particular Entity is connected into the overall topology of the audio function.

Input Terminal 1 (IT 1) could be the representation of a USB OUT endpoint used to stream audio from the Host to the audio device. IT 2 could be the representation of an analog Line-In connector on the audio device whereas IT 3 could be 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 (Volume, Bass, Treble …) before it is presented to Output Terminal 9 (OT 9). OT 9 could be the representation of a Headphone Out jack on the audio device.

At the same time, all three input sources (USB OUT, Line In, and Mic 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 could perform some audio processing algorithm(s) on the mix. The result is in turn sent to FU 8 where some final adjustments to the audio (Volume …) are made. FU 8 is connected to OT 10 and OT 11. OT 10 could represent speakers incorporated into the audio device and OT 11 could represent a USB IN endpoint used to send the processed audio to the Host for recording purposes.

Clock Source 12 (CS 12) could represent an internal sampling frequency generator, running at 96 kHz for instance. Clock Source 15 (CS 15) could be the representation of an external master sampling clock input that can be used to synchronize the device to an external source. Clock Selector 13 (CS 13) enables selection between the two available Clock Sources. The output of CS 13 provides a sampling frequency to IT 1, IT 2, IT3, OT 10, and OT 11 of 96 kHz. Clock Multiplier CM 14 further multiplies that clock signal by 0.5, providing 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 master clock (internal or external), 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.

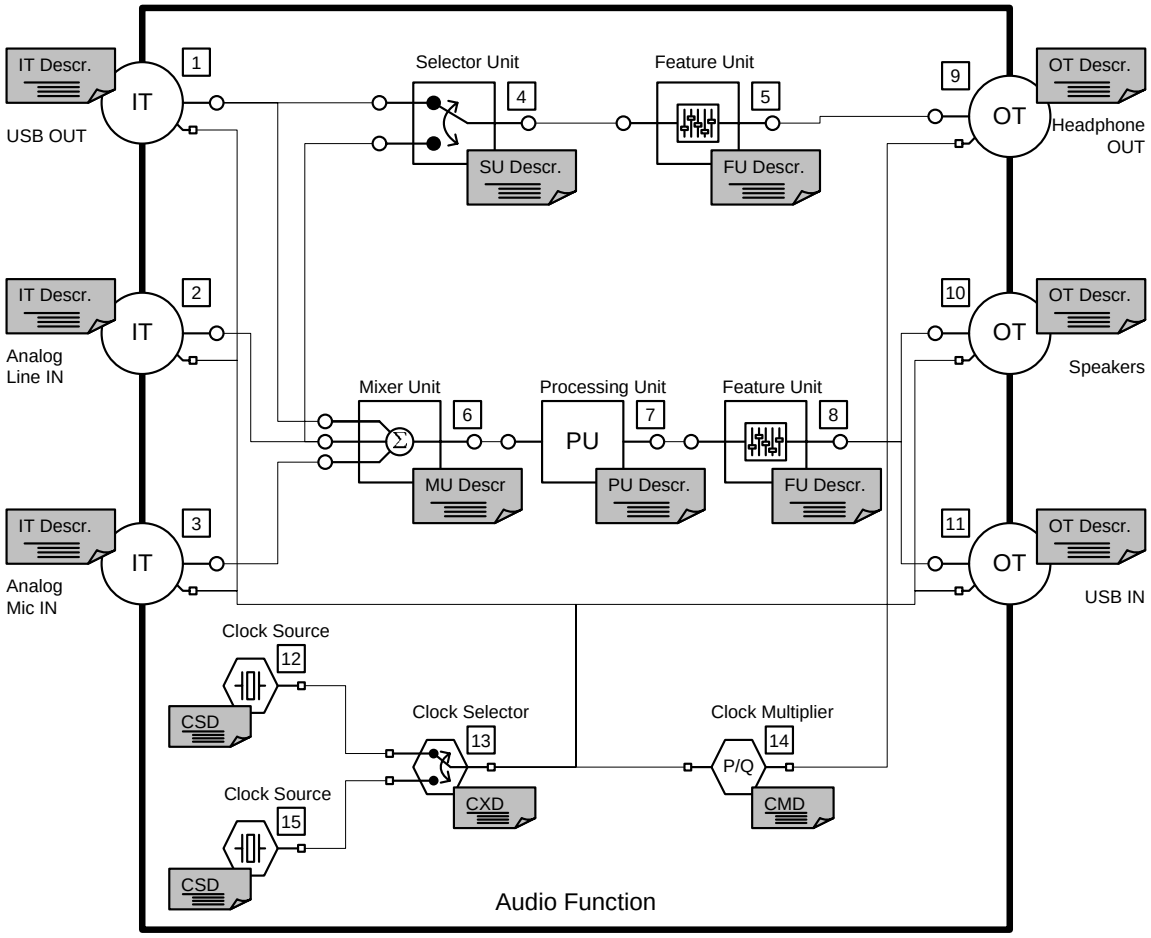

Figure 3-2: Inside the Audio Function

Inside an Entity, functionality is further described through Audio Controls. A Control typically provides access to a specific audio or clock property. Each Control has a set of attributes that can be manipulated or that present additional information on the behavior of the Control. A Control can have the following attributes:

- Current setting attribute
- Range attribute triplet consisting of:
  - Minimum setting attribute
  - Maximum setting attribute
  - Resolution attribute

As an example, consider a Volume Control inside a Feature Unit. By issuing the appropriate Get requests, the Host software can obtain values for the Volume Control's attributes and, for instance, use them to correctly display the Control on the screen. Setting the Volume Control's current attribute allows the Host software to change the volume setting of the Volume Control.

Additionally, each Entity in an audio function can have a memory space attribute. This attribute optionally provides generic access to the internal memory space of the Entity. This could be used to implement vendor-specific control of an Entity through generically provided access.

3.13.1 Audio Channel Cluster

An audio channel cluster is a grouping of audio channels that carry tightly related synchronous audio information. Inside the audio function, complete abstraction is made of the actual physical representation

form of the audio data that travels over the connections between Terminals and Units. Each audio channel in the cluster is considered to be a logical channel and all the physical attributes of the channel (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 together in the audio function's topology is a guarantee (by construction) that the information flowing over the connection is compatible with both Entities that are connected.

Channel numbering in the cluster starts with channel one up to the number of channels in the cluster. The virtual channel zero is used to address a master Control in a Unit (if present), effectively influencing all the channels at once. Note that the master Control (if present) must be implemented separate from the per-channel Controls. Changing the setting of a master Control does not affect the settings of any of the individual channel Controls.The maximum number of independent channels in an audio channel cluster is limited to 255 (Channel zero is used to reference the master channel).

In many cases, each channel in the audio cluster is tied to a certain location in the listening space. A trivial example of this is a cluster that contains Left and Right audio channels. To be able to describe more complex cases in a manageable fashion, this specification imposes some limitations and restrictions on the ordering of channels in an audio channel cluster.

To support the case where the information flowing over a connection cannot be interpreted as a cluster of logical audio channels, this specification defines an additional virtual spatial location called Raw Data. This spatial location is mutually exclusive with all other spatial locations defined in this specification. It cannot co-exist in the same cluster with other spatial locations and its use is very restricted. It can only be used on connections between Entities that do not manipulate audio content. These Entities can only be Input Terminals and Output Terminals. All other use of the Raw Data virtual spatial location is prohibited.

There are 28 predefined spatial locations (27 + 1 virtual):

- Front Left - FL
- Front Right - FR
- Front Center - FC
- Low Frequency Effects - LFE
- Back Left - BL
- Back Right - BR
- Front Left of Center - FLC
- Front Right of Center - FRC
- Back Center - BC
- Side Left - SL
- Side Right - SR
- Top Center - TC
- Top Front Left - TFL
- Top Front Center - TFC
- Top Front Right - TFR
- Top Back Left - TBL
- Top Back Center - TBC
- Top Back Right – TBR
- Top Front Left of Center – TFLC
- Top Front Right of Center – TFRC
- Left Low Frequency Effects – LLFE
- Right Low Frequency Effects– RLFE
- Top Side Left – TSL
- Top Side Right – TSR
- Bottom Center – BC
- Back Left of Center – BLC
- Back Right of Center – BRC
- Raw Data – RD

If there are channels present in the audio channel cluster that correspond to some of the previously defined spatial positions, then they must appear in the order specified in the above list. For instance, if a cluster contains channels Front Left, Front Right and LFE, then channel 1 is Front Left, channel 2 is Front Right, and channel 3 is LFE.

An audio channel cluster is characterized by only two attributes:

- The number of audio channels in the cluster
- The spatial location of each audio channel in the cluster, either predefined (Front Left, Front Right, Back Left, LFE, etc.) or custom-defined.

  Note: Custom-defined may also be used to specify channel assignments that are not directly related to any spatial location. A simple channel enumeration schema (Channel1, Channel2, etc.) is a typical example.

There is a Cluster Descriptor (CD) associated with each audio channel cluster that fully describes the cluster.

There are two types of audio channel clusters used in this specification:

- A logical audio channel cluster describes audio data within the audio function (the closed box) where audio channels are treated as logical concepts.
- A physical audio channel cluster describes audio data within an AudioStreaming interface that handles the actual physical audio channels (discrete or encoded) in the audio stream.

3.13.1.1 Mapping between Physical and Logical Audio Channel Clusters

AudioStreaming interfaces can have multiple Alternate Settings, each representing a different ‘mode of operation’ of the interface. For instance, an AudioStreaming interface, representing an external S/PDIF connection could have one Alternate Setting that is used when the S/PDIF connection is carrying Stereo PCM data (two channels) and another Alternate Setting that is used when AC-3 encoded information is conveyed over the S/PDIF connection (6 channels). Switching from one Alternate Setting to another can therefore effectively change both the number of actual audio channels in the interface and possibly also the spatial locations of those channels.

There is a one-to-one relationship between an AudioStreaming interface and its corresponding Terminal inside the audio function. However, the AudioStreaming interface can have multiple Alternate Settings with different physical audio channel clusters for each Alternate Setting. It must be possible to map each of these physical audio channel clusters onto the single logical audio channel cluster that is exposed via the corresponding Terminal. Therefore, the logical audio channel cluster is dynamic in nature and potentially changes whenever a different Alternate Setting is selected for the AudioStreaming interface. The current logical audio channel cluster can be retrieved through a Get Cluster Control request on the Terminal if implemented. Alternatively, Host software can keep track of which Alternate Setting is currently active and retrieve the physical audio channel cluster description from that Alternate Setting and use it as the logical channel cluster definition.

3.13.2 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 function 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 an audio channel cluster and leave the Input Terminal through a single Output Pin.

An Input Terminal can 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. However, if the audio stream is entering the audio function by means of a USB OUT endpoint, there is a one-to-one relationship between

the AudioStreaming interface that contains this endpoint and its associated Input Terminal. The class-specific interface descriptor contains a field that holds a direct reference to this Input Terminal. The Host needs to use both the AudioStreaming interface and endpoint descriptors and 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 descriptors. Control-related parameters are stored in the Terminal descriptor.

The conversion process from incoming, possibly encoded, audio streams to logical audio channels always involves some kind of decoding engine. This specification defines several types of decoding (See Section 3.14, “Encoders and Decoders.”) These decoding types range from rather trivial decoding schemes like converting interleaved stereo 16 bit PCM data into a Left and Right logical channel 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 actually receives the encoded audio data streams (like a USB AudioStreaming interface). The type of decoding is therefore implied in the **bmFormats** value, located in the class-specific AudioStreaming interface descriptor. Requests specific to the decoding engine must be directed to the AudioStreaming interface. The associated Input Terminal deals with the logical channels after they have been decoded.

An 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.

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

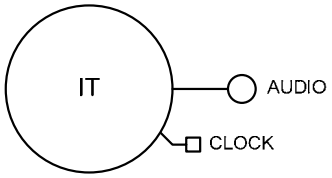

Figure 3-3: Input Terminal Icon

3.13.3 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 function 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 audio channel cluster enters the Output Terminal through a single Input Pin.

An Output Terminal can 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. However, if the audio stream is leaving the audio function by means of a USB IN endpoint, there is a one-to-one relationship between the AudioStreaming interface that contains this endpoint and its associated Output Terminal. The class-specific interface descriptor contains a field that holds a direct reference to this Output Terminal. The Host needs to use both the AudioStreaming interface and endpoint descriptors and the Output Terminal descriptor to fully understand the characteristics and capabilities of the Output Terminal. Stream-related parameters are stored in the AudioStreaming descriptors. Control-related parameters are stored in the Terminal descriptor.

The conversion process from incoming logical audio channels to possibly encoded audio streams always involves some kind of encoding engine. This specification defines several types of encoding (see Section 3.14, “Encoders and Decoders,”) ranging from rather trivial to very sophisticated schemes. The encoding engine is considered part of the Entity that actually transmits the encoded audio data streams (like a USB AudioStreaming interface). The type of encoding is therefore implied in the **bmFormats** value, located in the class-specific AudioStreaming interface descriptor. Requests specific to the encoding engine must be

directed to the AudioStreaming interface. The associated Output Terminal deals with the logical channels before encoding.

An 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.

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

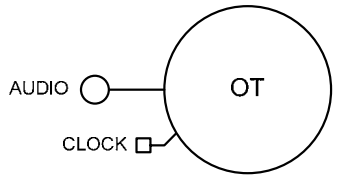

Figure 3-4: Output Terminal Icon

3.13.4 Mixer Unit

The Mixer Unit (MU) transforms a number of logical input channels into a number of logical output channels. The input channels are grouped into one or more audio channel clusters. Each cluster enters the Mixer Unit through an Input Pin. The logical output channels are grouped into one audio channel cluster and leave the Mixer Unit through a single Output Pin.

Every input channel can virtually be mixed into all of the output channels. If n is the total number of input channels and m is the number of output channels, then there is a two-dimensional array (n · m) of Mixer Controls in the Mixer Unit. Not all of these Controls have to be physically implemented. Some Controls can have a fixed setting and be non-programmable. The Mixer Unit Descriptor reports which Controls are programmable in the **bmControls** bitmap field. Using this model, a permanent connection can be implemented by reporting the Control as non-programmable in the bmControls bitmap and by returning a Control setting of 0 dB when requested. Likewise, a missing (muting) connection can be implemented by reporting the Control as non-programmable in the bmControls bitmap and by returning a Control setting of –∞ dB. A Mixer Unit MUST respond to the appropriate Get Request to allow the Host to retrieve the actual settings on each of the (n · m) Mixer Controls.

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

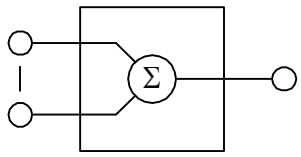

Figure 3-5: Mixer Unit Icon

3.13.5 Selector Unit

The Selector Unit (SU) selects from n audio channel clusters, each containing m logical input channels and routes them unaltered to the single output audio channel cluster, containing m output channels. It represents a multi-channel source selector, capable of selecting between n m-channel sources. It has n Input Pins and a single Output Pin.

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

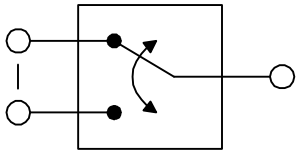

Figure 3-6: Selector Unit Icon

3.13.6 Feature Unit

The Feature Unit (FU) is essentially a multi-channel processing unit that provides basic manipulation of multiple single-parameter Audio Controls on the incoming logical channels. For each logical channel, the Feature Unit optionally provides Audio Controls for the following features:

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

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

The logical channels in the cluster are numbered from one to the total number of channels in the cluster. The ‘master’ channel has channel number zero and is always virtually present.

The Feature Unit Descriptor reports what Controls are present for every channel in the Feature Unit and for the ‘master’ channel. All logical channels in a Feature Unit are fully independent. There exist no cross couplings among channels within the Feature Unit. There are as many logical output channels, as there are input channels. These are grouped into one audio channel cluster that enters the Feature Unit through a single Input Pin and leaves the Unit through a single Output Pin.

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

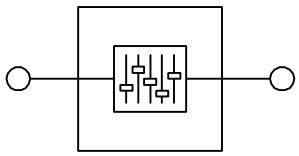

Figure 3-7: Feature Unit Icon

3.13.7 Sampling Rate Converter Unit

The Sampling Rate Converter Unit 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 can 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 Sampling Rate Converter Unit (RU) provides a bridge function between different clock domains within the audio function. The Sampling Rate Converter (SRC) does not provide any Audio Controls. It takes the audio on all the logical channels in the single input audio cluster belonging to a certain clock domain and converts them into the same logical channels in the single output cluster but now belonging to another clock domain.

There are as many logical output channels, as there are input channels. These are grouped into one audio channel cluster that enters the SRC Unit through a single Input Pin and leaves the Unit through a single Output Pin.

A SRC Unit has a 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.

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

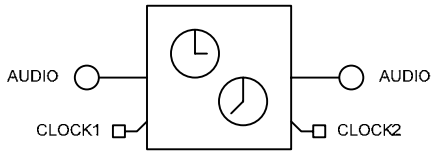

Figure 3-8: Sampling Rate Converter Unit Icon

3.13.8 Effect Unit

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

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

In addition, the Effect Unit optionally provides one of the above Audio Controls but now influencing all channels of the cluster at once. In this way, a ‘master’ Control can be implemented. The master Control is cascaded after the individual channel Controls. This setup is especially useful in multi-channel systems where the individual channel Controls can be used for channel balancing and the master Control can be used for overall settings.

The logical channels in the cluster are numbered from one to the total number of channels in the cluster. The ‘master’ channel has channel number zero and is always virtually present.

The Effect Unit Descriptor reports what Controls are present for every channel in the Effect Unit and for the ‘master’ channel. All logical channels in an Effect Unit are fully independent. There exist no cross couplings among channels within the Effect Unit. There are as many logical output channels, as there are input channels. These are grouped into one audio channel cluster that enters the Effect Unit through a single Input Pin and leaves the Unit through a single Output Pin.

3.13.8.1 Parametric Equalizer Section Effect Unit

The Parametric Equalizer Section (PEQS) Effect Unit is used to manipulate and equalize the frequency characteristics of the original audio information around a certain center frequency. In order 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 master channel concept allows equalization for all channels simultaneously.

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

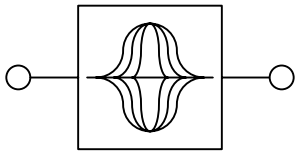

Figure 3-9: PEQS Effect Unit Icon

3.13.8.2 Reverberation Effect Unit

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

- Reverb Type: Room1, Room2, Room3, Hall1, Hall2, Plate, Delay, and Panning Delay.
- 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 symbol for the Reverberation Effect Unit can be found in the following figure:

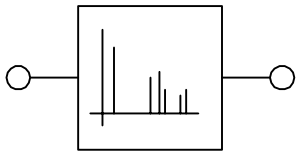

Figure 3-10: Reverberation Effect Unit Icon

3.13.8.3 Modulation Delay Effect Unit

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

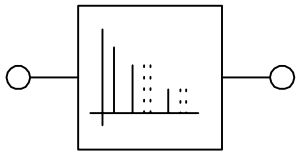

Figure 3-11: Modulation Delay Effect Unit Icon

3.13.8.4 Dynamic Range Compressor Effect Unit

The Dynamic Range Compressor Effect Unit is used to intelligently limit the dynamic range of the original audio information. A number of parameters can be manipulated to influence the desired compression.

Error! Objects cannot be created from editing field codes.

Figure 3-12: Dynamic Range Compressor Transfer Characteristic

- Compression ratio R: determines the slope of the static input-to-output transfer characteristic in the compressor's active input range. The compression is defined in terms of the compression 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.

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

  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 compressor 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 a loud passage. Expressed in ms.
- Make-up Gain: set to compensate for the gain loss in the compressor. Expressed in dB

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

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

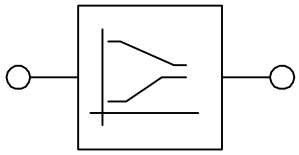

Figure 3-13: Dynamic Range Compressor Effect Unit Icon

3.13.9 Processing Unit

The Processing Unit (PU) represents a functional block inside the audio function that transforms a number of logical input channels, grouped into one or more audio channel clusters into a number of logical output channels, grouped into one audio channel cluster. Therefore, the Processing Unit can have multiple Input Pins and has a single Output Pin. 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.

Processing Units are encouraged to support at least the Enable Control, allowing the Host software to bypass whatever functionality is incorporated in the Processing Unit.

3.13.9.1 Up/Down-mix Processing Unit

The Up/Down-mix 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 can be proprietary. The input channels are grouped into one input channel cluster that enters the Processing Unit over a single Input Pin. Likewise, all output channels are grouped into one output channel cluster, leaving the Processing Unit over a single Output Pin.

The Up/Down-mix Processing Unit can support multiple modes of operation (besides the bypass mode, controlled by the Enable Control). The available input audio channels are dictated by the Unit or Terminal to which the Up/Down-mix Processing Unit is connected. The Up/Down-mix Processing Unit descriptor reports which up/down-mixing modes the Unit supports through its **waModes()** array. Each element of the **waModes()** array indicates which output channels in the output cluster are effectively used in a particular mode. The unused output channels in the output cluster must produce muted output. Mode selection is implemented using the Get/Set Control request.

As an example, consider the case where an Up/Down-mix Processing Unit is connected to an Input Terminal, producing Dolby™ AC-3 5.1 decoded audio. The input audio channel cluster to the Up/Down-mix Processing Unit therefore contains Left, Right, Center, Left Surround, Right Surround 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 audio function's discretion 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 routed. For instance, a different down-mix algorithm is needed whether the 'enriched' stereo stream is sent to a pair of speakers or to a headphone set. 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 instance 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 (Left of Center, 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.

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

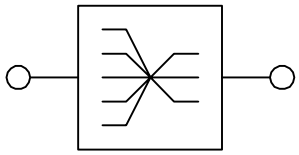

Figure 3-14: Up/Down-mix Processing Unit Icon

3.13.9.2 Dolby Prologic Processing Unit

The Dolby Prologic™ decoding process can be seen as an operator on the Left and Right logical channels of the input cluster of the Unit. It is capable of extracting additional audio data (Center and/or Surround channels) from information that is transparently 'superimposed' on the Left and Right audio channels. It therefore differs from a true decoding process as defined for an Input Terminal. It can be applied on a logical audio stream anywhere in the audio function. The Dolby Prologic Processing Unit is a specialized derivative of the Up/Down-mix Processing Unit.

The Dolby Prologic Processing Unit can have the following modes of operation (besides the bypass mode, controlled by the Enable Control):

- Left, Right, Center channel decoding
- Left, Right, Surround channel decoding
- Left, Right, Center, Surround decoding

The Dolby Prologic Processing Unit descriptor reports which modes the Unit supports. Mode selection is then implemented using the Get/Set Control request.

Dolby Prologic Surround Delay Control is considered not to be part of the Dolby Prologic™ Processing Unit and must be handled by a separate Feature Unit.

Dolby Prologic Bass Management is the local responsibility of the audio function and should not be controllable from the Host.

The symbol for the Dolby Prologic Processing Unit can be found in the following picture:

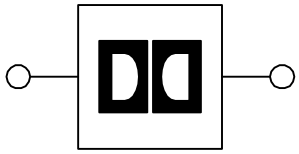

Figure 3-15: Dolby Prologic Processing Unit Icon

3.13.9.3 Stereo Extender Processing Unit

The Stereo Extender Processing Unit operates on Left and Right channels only. It processes an existing stereo (two channel) soundtrack to expand the sound field and to make it appear to originate from outside the Front Left/Right speaker locations. Extended stereo effects can be achieved via various methods. The algorithms and transforms applied to accomplish this are not defined by this specification and can be proprietary. The effects of the Stereo Extender Processing Unit can be bypassed at all times through manipulation of the Enable Control. The perceived width of the sound field can be controlled using the proper Get/Set Control request.

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

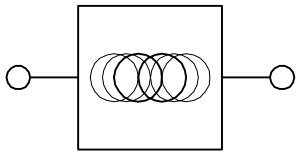

Figure 3-16: Stereo Extender Processing Unit Icon

3.13.10 Extension Unit

The Extension Unit (XU) is the method provided by this specification to easily add vendor-specific building blocks to the specification. The Extension Unit provides one or more logical input channels, grouped into one or more audio channel clusters and transforms them into a number of logical output

channels, grouped into one audio channel cluster. Therefore, the Extension Unit can have multiple Input Pins and has a single Output Pin.

Extension Units are required to support at least the Enable Control, allowing the Host software to bypass whatever functionality is incorporated in the Extension Unit.

Although a generic audio driver will not be able to determine what functionality is implemented in the Extension Unit, let alone manipulate it, it still will be capable of recognizing the presence of vendor-specific extensions and assume default behavior for those units.

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

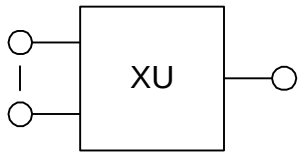

Figure 3-17: Extension Unit Icon

3.13.11 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 within the audio function. A Terminal inside the audio function can only be connected to one Clock Entity. The clock signal present at the Clock Input Pin of a Terminal determines the sampling frequency at which the underlying hardware is operating.

3.13.11.1 Clock Source

A Clock Source Entity provides an independent sampling clock signal on its single Clock Output Pin. The Clock Source Entity serves as the master clock for a Clock Domain. Several different other synchronous sampling frequencies can be derived by connecting multiple Clock Multiplier Entities to the same Clock Source Entity. By manipulating the Sampling Frequency Control inside the Clock Source Entity, all sampling frequencies in the Clock domain are influenced. Changing the Sampling Frequency Control value is the only way provided by this specification to change the sampling clock.

Each independent master sampling clock inside the audio function must be represented by a separate Clock Source Entity. 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 must 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 origin of the actual clock signal, represented by the Entity. Furthermore, since Input and Output Terminals only have a Clock Input Pin, a clock signal can never be generated from a Terminal directly.

The output of a Clock Source Entity does not have to be valid at all times. For instance, if a Clock Source Entity represents an external sampling clock input on the audio function, the output of that Clock Source might not be valid when there is nothing connected to the external clock input. The Clock Source can be queried for the validity of its output signal at all times.

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

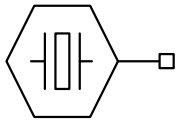

Figure 3-18: Clock Source Icon

3.13.11.2 Clock Selector

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

Switching between Clock Inputs can be Host controlled (the Clock Selector is programmable via the appropriate Set request) or the audio function can switch Clock Inputs due to some external event. A Clock Selector may support both control methods. The Clock Selector can notify the Host of the change by generating the appropriate interrupt.

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

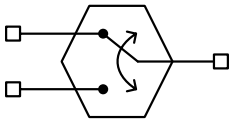

Figure 3-19: Clock Selector Icon

3.13.11.3 Clock Multiplier

A Clock Multiplier Entity provides the functionality to derive a new sampling clock signal frequency from the sampling clock signal, present at its single Clock Input Pin. The algorithm used to derive the new clock signal is not defined by this specification. However, there is a requirement that the output clock signal must be synchronous to the input clock signal so that both clocks belong to the same Clock Domain. A Clock Multiplier Entity contains a multiplier followed by a divider.Both the multiplication factor P and division factor Q are fixed (read-only) and in the range [1, 2^{16}-1]. The resulting sampling frequency is obtained by multiplying the input signal frequency by P/Q.

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

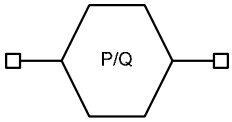

Figure 3-20: Clock Multiplier Icon

3.14 Encoders and Decoders

Whenever an audio data stream is entering or leaving the audio function, there is some sort of decoding or encoding process involved. This process can be fairly trivial (interleaving or de-interleaving samples) but it can also be quite elaborate.

In some cases, interaction between Host software and the encoding or decoding process is necessary. At this time five encoder and decoder processes of this type are defined:

- MPEG Encoder and Decoder
- AC-3 Encoder and Decoder
- Windows Media Audio (WMA) Encoder and Decoder
- DTS Encoder and Decoder
- OTHER Encoder and Decoder

These processes each have their specific Controls defined that allow manipulation or monitoring of the internal status of the encoder or decoder processes. These Controls are addressed through the AudioStreaming interface to which the encoder or decoder belongs.

3.15 Copy Protection

Because the Audio Device Class is primarily dealing with digital audio streams, the issue of protecting these – often-copyrighted – streams cannot be ignored. Therefore, this specification provides the means to preserve whatever copyright information is available. However, it is the responsibility of the Host software to manage the flow of copy protection information throughout the audio function.

Copy protection issues come into play whenever digital audio streams enter or leave the audio function. Therefore, the copy protection mechanism is implemented at the Terminal level in the audio function. Streams entering the audio function can be accompanied by specific information, describing the copy protection level of that audio stream. Likewise, streams leaving the audio function should be accompanied by the appropriate copy protection information, if the hardware permits it. This specification provides for two dedicated requests that can be used to manage the copy protection mechanism. The Get Copy Protect request can be used to retrieve copy protection information from an Input Terminal whereas the Set Copy Protect request is used to preset the copy protection level of an Output Terminal.

This specification provides for three levels of copy permission, similar to CGMS (Copy Generation Management System) and SCMS (Serial Copy Management System).

- Level 0: Copying is permitted without restriction. The material is either not copyrighted, or the copyright is not asserted.
- Level 1: One generation of copies may be made. The material is copyright protected and is the original.
- Level 2: The material is copyright protected and no digital copying is permitted.

3.16 Operational Model

A device can support multiple configurations. Within each configuration can be multiple interfaces, each possibly having Alternate Settings. These interfaces can pertain to different functions that co-reside in the same composite device. Even several independent audio functions can exist in the same device. Interfaces, belonging to the same audio function are grouped into an Audio Interface Collection. If the device contains multiple independent audio functions, there must be multiple Audio Interface Collections, each providing full access to their associated audio function.

As an example of a composite device, consider a PC monitor 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 a Collection 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 (Volume 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 could duplicate this audio stream into two logical channels, and those could then be reproduced on both speakers. From an interface point of view, such a setup requires one isochronous endpoint in 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 must be 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 now splits the stream into a Front Left and Front Right logical channel. 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 Collection, 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.

3.16.1 AudioControl Interface

To control the functional behavior of a particular audio function, the Host can manipulate the Clock Entities, Units and Terminals inside the audio function. To make these objects accessible, the audio function must expose a single AudioControl interface. This interface can contain the following endpoints:

- A control endpoint for manipulating Clock Entity, Unit and Terminal 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. This endpoint is optional but extremely useful and should be implemented on all but the simplest audio devices so that Host software can be notified at all times of any change in the audio function's behavior.

The AudioControl interface is the single entry point to access the internals of the audio function. All requests that are concerned with the manipulation of certain Audio Controls within the audio function's Clock Entities, Units or Terminals must 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 can only support a single Alternate Setting (Alternate Setting 0).

3.16.1.1 Control Endpoint

The audio interface class uses endpoint 0 (the default pipe) as the standard way to control the audio function using class-specific requests. These requests are always directed to one of the Units or Terminals that make up the audio function. The format and contents of these requests are detailed further in this document.

3.16.1.2 Interrupt Endpoint

A USB AudioControl interface can support an optional interrupt endpoint to inform the Host about dynamic changes that occur on the different addressable Entities (Clock Entities, Terminals, Units, interfaces and endpoints) inside the audio function. The interrupt endpoint is used by the entire Audio Interface Collection to convey change information to the Host. It is considered part of the AudioControl interface because this is the anchor interface for the Collection.

3.16.2 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 can have zero or more AudioStreaming interfaces associated with it, each possibly carrying data of a different nature and format. Each AudioStreaming interface can 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 must follow the endpoint numbering scheme as set forth in the *USB Specification*, Section 9.6.6, “Endpoint”

An AudioStreaming interface can have Alternate Settings that can be used to change certain characteristics of the interface and 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 must at least provide the default Alternate Setting (Alternate Setting 0) with zero bandwidth requirements (no isochronous data endpoint defined) and an additional Alternate Setting that contains the actual isochronous data endpoint.

The AudioStreaming interface is essentially used to provide an access point for the Host software (drivers) to manipulate the behavior of the physical interface it represents. Therefore, even external connections to the audio function (S/PDIF interface, analog input, etc.) can be represented by an AudioStreaming interface so that the Host software can control certain aspects of those connections. This type of AudioStreaming interface has no associated USB endpoints. The related audio data stream is not using USB as a transport medium.

For every isochronous OUT or IN endpoint defined in any of the AudioStreaming interfaces, there must 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 must take into account the interface- and endpoint-related descriptors as well as the Terminal-related descriptor.

3.16.2.1 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 can be entirely different from the native 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 must convert logical channels from the Output Terminal into an audio data stream, often using some form of encoding. If the decoding or encoding process exposes Controls that influence the encoding or decoding, these Controls can be accessed through the AudioStreaming interface.

Requests to control properties that exist within an audio function, such as volume 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 requests must be directed to the proper audio function’s Units or Terminals via the AudioControl interface.

As already mentioned, an AudioStreaming interface can have zero or one isochronous audio data endpoint. If multiple synchronous audio channels must be communicated between Host and audio function, they must be clustered into one physical audio channel 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 Collection (all servicing the same audio function).

3.16.2.2 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, see Section 5, “USB Data Flow Model,” in the *USB Specification*. For specific information on the formatting

of the data going over the feedback pipe, refer to Section 5.12.4.2, “Feedback” and Section 9.6.6, “Endpoint” in the *USB Specification*.

3.16.2.3 Audio Data Format

The format used to transport audio data over the USB is entirely determined by the codes, located in the **bFormatType** and **bmFormats** fields of the class-specific interface descriptor.For each defined Format Type, a Format Type descriptor is needed to fully describe the format. For details about the defined Format Types and associated data formats and descriptors, see the separate document, *USB Audio Data Formats* that is considered part of this specification. Vendor-specific protocols must be fully documented by the manufacturer.

3.16.3 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 can only be influenced by directly interacting with the Sampling Frequency Control inside a Clock Source Entity. The Sampling Frequency Controls RANGE attributes provide the necessary information for Host software to determine what sampling frequencies the Control (and thus the associated Clock Domain) 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 must 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 request 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. Instead, it is probably enough to provide a single Active Alternate Setting if the Host software is capable of bandwidth reclamation (even on isochronous transfers). Alternatively, just a few Active Alternate Settings (medium bandwidth, high bandwith) might be enough to provide reasonable bandwidth control.

Audio streams can be bridged from one clock domain to another through the use of the Sampling Rate Converter Unit.

3.16.4 Binding between Physical Buttons and Audio Controls

Most devices that contain an audio function will also have one or more frontpanel buttons that are intended to control certain aspects of the audio function inside the device. The most obvious example is a Volume Control button on the front of a multimedia speaker. Since an audio function can potentially contain many Audio Controls of the same type, there is a need to bind a physical control (button, knob, slider, jog, …) to a particular Audio Control inside the audio function.

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

- The physical button is implemented as a HID Control
- The physical button is an integral part of the Audio Control

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 frontpanel buttons and the second method for the remaining frontpanel buttons. It is strongly discouraged to implement frontpanel buttons that use neither of the above mentioned methods, i.e. buttons that are invisible to Host software and have a local effect only.

3.16.4.1 Physical button is a HID Control

In this case, the physical button is completely separate from the audio function and is implemented within the device's HID interface. The audio function is not even aware of the button's existence. 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 derive from there 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.

3.16.4.2 Physical button is Integral Part of the Audio Control

In this case, the physical button directly interacts with the actual Audio Control. Button state changes are not reported to Host software. Instead, the change of state of the Audio Control resulting from the button manipulation is reported to Host software through the Audio Control interrupt mechanism. As a consequence, the binding between the physical button and the Audo Control is very direct and entirely dictated by the design of the device. Although less flexible, this method provides a very clear and straightforward way to perform the binding.

4 Descriptors

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

4.1 Audio Channel Cluster Descriptor

An audio channel cluster is a grouping of audio channels that share the same characteristics like sampling frequency, bit resolution, etc. To characterize an audio channel cluster, a cluster descriptor is introduced.

The audio channel cluster descriptor contains the following fields:

- **bNrChannels**: a number that specifies how many audio channels are present in the cluster.
- **bmChannelConfig**: a bitmap field that indicates which spatial locations are occupied by the channels present in the cluster. The bit allocations are as follows:
  - D0: Front Left - FL
  - D1: Front Right - FR
  - D2: Front Center - FC
  - D3: Low Frequency Effects- LFE
  - D4: Back Left - BL
  - D5: Back Right - BR
  - D6: Front Left of Center - FLC
  - D7: Front Right of Center - FRC
  - D8: Back Center - BC
  - D9: Side Left - SL
  - D10: Side Right - SR
  - D11: Top Center - TC
  - D12: Top Front Left - TFL
  - D13: Top Front Center - TFC
  - D14: Top Front Right - TFR
  - D15: Top Back Left - TBL
  - D16: Top Back Center - TBC
  - D17: Top Back Right – TBR
  - D18: Top Front Left of Center – TFLC
  - D19: Top Front Right of Center – TFRC
  - D20: Left Low Frequency Effects– LLFE
  - D21: Right Low Frequency Effects– RLFE
  - D22: Top Side Left – TSL
  - D23: Top Side Right – TSR
  - D24: Bottom Center – BC
  - D25: Back Left of Center – BLC
  - D26: Back Right of Center – BRC
  - D27..D30: Reserved
  - D31: Raw Data – RD; Mutually exclusive with all other spatial locations

  Each bit set in this bit map indicates there is a channel in the cluster that carries audio information, destined for the indicated spatial location. The channel ordering in the cluster must correspond to the ordering imposed by the above list of predefined spatial locations. If there are more channels in the cluster than there are bits set in the **bmChannelConfig** field, (i.e. **bNrChannels > [**Number_Of_Bits_Set]), then the first [Number_Of_Bits_Set] channels take the spatial positions, indicated in **bmChannelConfig.** The remaining channels have ‘non-predefined’ spatial positions (positions that do not appear in the predefined list). If none of the bits in **bmChannelConfig** are set, then all channels have non-predefined spatial positions. If one or more channels have non-predefined

spatial positions, their spatial location description can optionally be derived from the **iChannelNames** field.

- **iChannelNames**: index to a string descriptor that describes the spatial location of the first non-predefined logical channel in the cluster. The spatial locations of all remaining logical channels **must** be described by string descriptors with indices that immediately follow the index of the descriptor of the first non-predefined channel. Therefore, **iChannelNames** inherently describes an array of string descriptor indices, ranging from **iChannelNames** to (**iChannelNames** + (**bNrChannels**-[Number_Of_Bits_Set]) - 1)

In the special case where the cluster is carrying Raw Data, the **bNrChannels** field must be set to zero, the **bmChannelConfig** bitmap must have only bit D31 set, and the **iChannelNames** field must also be set to zero. When interpreting the cluster descriptor, all other information in the cluster descriptor must be ignored when bit D31 in the **bmChannelConfig** field is set.

Table 4-1: Audio Channel Cluster Descriptor

Offset	Field	Size	Value	Description
0	bNrChannels	1	Number	Number of logical output channels in the Terminal's output audio channel cluster.
1	bmChannelConfig	4	Bitmap	Describes the spatial location of the logical channels.
5	iChannelNames	1	Index	Index of a string descriptor, describing the name of the first logical channel.

Example 1:

An audio channel cluster that carries Dolby Prologic channels has the following cluster descriptor:

Table 4-1: Dolby Prologic Cluster Descriptor

Offset	Field	Size	Value	Description
0	bNrChannels	1	4	There are 4 channels in the cluster.
1	bmChannelConfig	4	0x00000107	Front Left, Front Right, Front Center and Back Center are present.
5	iChannelNames	1	Index	Because there are no non-predefined channels, this index must be set to 0.

Example 2:

A hypothetical audio channel cluster inside an audio function could carry Left, Left Surround, Left of Center, and two auxiliary channels that contain each a different weighted mix of the Left, Left Surround and Left of Center channels. The corresponding cluster descriptor would be:

Table 4-2: Left Group Cluster Descriptor

Offset	Field	Size	Value	Description
0	bNrChannels	1	5	There are 5 channels in the cluster

Offset	Field	Size	Value	Description
1	bmChannelConfig	4	0x00000051	Front Left, Back Left , Front Left of Center and two undefined channels are present. (bNrChannels > [Number_Of_Bits_Set])
5	iChannelNames	1	Index	Optional index of the first non-predefined string descriptor

Optional string descriptors:

```
String (Index)   = 'Left Down Mix 1'
String (Index+1) = 'Left Down Mix 2'
```

This specification makes a distinction between a logical and physical audio channel cluster. Therefore there are also two types of channel cluster descriptors defined:

- Logical audio channel cluster descriptor
- Physical audio channel cluster descriptor

The layout of these descriptors is identical. Both the logical and physical cluster descriptors are not stand-alone descriptors as such. The logical audio channel cluster descriptor is embedded within one of the following descriptors:

- Input Terminal descriptor
- Mixer Unit descriptor
- Processing Unit descriptor
- Extension Unit descriptor

The physical audio channel cluster descriptor is always embedded within the class-specific AS interface descriptor.

Since logical audio channel clusters can be dynamic in nature, the logical channel cluster descriptors must contain the initial values, describing the cluster immediately after startup of the audio function. In many cases, this means that the logical audio channel descriptor will represent an empty cluster (all fields zero) since all AudioStreaming interfaces that represent a USB endpoint are required to start up in idle mode (Alternate Setting 0, zero bandwidth).

4.2 Device Descriptor

Because audio functionality is always considered to reside at the interface level, this class specification does not define a specific audio device descriptor. For both composite devices and audio-only devices, the device descriptor must indicate that class information is to be found at the interface level. Therefore, the **bDeviceClass, bDeviceSubClass** and **bDeviceProtocol** fields of the device descriptor must contain the values 0xEF, 0x02, and 0x01 respectively 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. All other fields of the device descriptor must comply with the definitions in Section 9.6.1, "Device" of the *USB Specification*. There is no class-specific Device descriptor.

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.

4.3 Device_Qualifier Descriptor

The Device_Qualifier descriptor obeys the same rules as the Device descriptor. Therefore, the **bDeviceClass, bDeviceSubClass** and **bDeviceProtocol** fields of the Device_Qualifier descriptor must also

be set to 0xEF, 0x02, and 0x01 respectively. All other fields of the Device_Qualifier descriptor must comply with the definitions in Section 9.6.2, “Device_Qualifier” of the *USB Specification*. There is no class-specific Device_Qualifier descriptor.

4.4 Configuration Descriptor

In analogy to the device descriptor, an audio configuration descriptor is applicable only in the case of audio-only devices. It is identical to the standard configuration descriptor defined in Section 9.6.3, “Configuration” of the *USB Specification*. There is no class-specific Configuration descriptor.

4.5 Other_Speed_Configuration Descriptor

The Other_Speed_Configuration descriptor is identical to the standard Other_Speed_Configuration descriptor defined in Section 9.6.4, “Other_Speed_Configuration” of the *USB Specification*. There is no class-specific Other_Speed_Configuration descriptor.

4.6 Interface Association Descriptor

The standard Interface Association mechanism is used to describe an Audio Interface Collection. All interfaces belonging to the same Audio Interface Collection must be identified by means of the standard Interface Association descriptor. The mandatory AudioControl interface must be the first in the collection (having the lowest interface number). All AudioStreaming interfaces must be contiguously numbered and immediately follow the AudioControl interface. All MIDIStreaming interfaces must be contiguously numbered and immediately follow the AudioStreaming interfaces.

The Interface Association Descriptor is defined in the *Interface Association Descriptor Engineering Change Notic*e (ECN) and must be considered an integral part of the *USB Specification*.

Table 4-3: Standard Interface Association Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	Number	Size of this descriptor in bytes: 8
1	bDescriptorType	1	Constant	INTERFACE ASSOCIATION Descriptor.
2	bFirstInterface	1	Number	Interface number of the first interface that is associated with this function.
3	bInterfaceCount	1	Number	Number of contiguous interfaces that are associated with this function.
4	bFunctionClass	1	Class	AUDIO_FUNCTION Function Class code (assigned by this specification). See Appendix A.1, “Audio Function Class Code”.
5	bFunctionSubClass	1	SubClass	FUNCTION_SUBCLASS_UNDEFINED Function Subclass code. Currently not used. See Appendix A.2, “Audio Function Subclass Codes”.
6	bFunctionProtocol	1	Protocol	AF_VERSION_02_00 Function Protocol code. Indicates the current version of the specification. See Appendix A.3, “Audio Function Protocol Codes”

Offset	Field	Size	Value	Description
7	iFunction	1	Index	Index of a string descriptor that describes this interface.

4.7 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 descriptor provides pertinent information concerning the internals of the audio function. It specifies revision level information and lists the capabilities of each Unit and Terminal.

4.7.1 Standard AC Interface Descriptor

The standard AC interface descriptor is identical to the standard interface descriptor defined in Section 9.6.5, "Interface" of the *USB Specification*, except that some fields have now dedicated values.

Table 4-4: 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.
3	bAlternateSetting	1	Number	Value used to select an Alternate Setting for the interface identified in the prior field. Must 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	Class	AUDIO. Audio Interface Class code (assigned by the USB). See Appendix A.4, "Audio Interface Class Code."
6	bInterfaceSubClass	1	Subclass	AUDIOCONTROL. Audio Interface Subclass code. Assigned by this specification. See Appendix A.5, "Audio Interface Subclass Codes."
7	bInterfaceProtocol	1	Protocol	IP_VERSION_02_00 Interface Protocol code. Indicates the current version of the specification. See Appendix A.6, "Audio Interface Protocol Codes"
8	iInterface	1	Index	Index of a string descriptor that describes this interface.

4.7.2 Class-Specific AC Interface Descriptor

The class-specific AC interface descriptor is a concatenation of all the descriptors that are used to fully describe the audio function, i.e. all Clock Descriptors (CDs), all Unit Descriptors (UDs), and Terminal Descriptors (TDs).

The total length of the class-specific AC interface descriptor depends on the number of Clock Entities, Units and Terminals in the audio function. Therefore, the descriptor starts with a header that reflects the total length in bytes of the entire class-specific AC interface descriptor in the **wTotalLength** field. The **bcdADC** field identifies the release of the Audio Device Class Specification with which this audio function and its descriptors are compliant. The **bCategory** field contains a constant that indicates what the primary use of this audio function is as intended by the manufacturer.

The order in which the Clock Entity, Unit, and Terminal descriptors are reported is not important because every descriptor can be identified through its **bDescriptorType** and **bDescriptorSubtype** field. The **bDescriptorType** field identifies the descriptor as being a class-specific interface descriptor. The **bDescriptorSubtype** field further qualifies the exact nature of the descriptor.

The **bmControls** field contains a set of bit pairs, indicating which Controls are present at the audio function level (as opposed to at the addressable entity level) and what their capabilities are. If a Control is present, it must be Host readable. If a certain Control is not present then the bit pair must be set to 0b00. If a Control is present but read-only, the bit pair must be set to 0b01. If a Control is also Host programmable, the bit pair must be set to 0b11. The value 0b10 is not allowed.

The following table defines the class-specific AC interface header descriptor.

Table 4-5: Class-Specific AC Interface Header Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	Number	Size of this descriptor, in bytes: 9
1	bDescriptorType	1	Constant	CS_INTERFACE descriptor type.
2	bDescriptorSubtype	1	Constant	HEADER descriptor subtype.
3	bcdADC	2	BCD	Audio Device Class Specification Release Number in Binary-Coded Decimal.
5	bCategory	1	Constant	Constant, indicating the primary use of this audio function, as intended by the manufacturer. See Appendix A.7, “Audio Function Category Codes.”
6	wTotalLength	2	Number	Total number of bytes returned for the class-specific AudioControl interface descriptor. Includes the combined length of this descriptor header and all Clock Source, Unit and Terminal descriptors.
8	bmControls	1	Bitmap	D1..0: Latency Control D7..2: Reserved. Must be set to 0.

This header is followed by the Clock Entity descriptors, the Unit descriptors and the Terminal descriptors. The order in which they appear is not important. The layout of the descriptors depends on the type of Clock Entity, Unit or Terminal they represent. There are three types of Clock Entity descriptors, six types of Unit descriptors, and two types of Terminal descriptors possible. They are summarized in the following sections.

The first four fields are common for all descriptors. They contain the Descriptor Length, Descriptor Type, Descriptor Subtype, and Clock Entity ID, Unit ID, or Terminal ID.

Each Clock Entity, Unit and Terminal within the audio function is assigned a unique identification number, the Clock Entity ID (CID), Unit ID (UID) or Terminal ID (TID), contained in the **bClockID**, **bUnitID** or **bTerminalID** field of the descriptor. The value 0x00 is reserved for undefined ID, effectively restricting the total number of addressable Entities in the audio function (Clock Entities, Units, and Terminals) to 255.

Besides uniquely identifying all addressable Entities in an audio function, the IDs also serve to describe the topology of the audio function; i.e. the **bSourceID** field of a Unit or Terminal descriptor indicates to which other Unit or Terminal this Unit or Terminal is connected. Likewise, the bCSourceID field in a Terminal descriptor indicates to which Clock Entity this Terminal is connected.

4.7.2.1 Clock Source Descriptor

The Clock Source Entity is uniquely identified by the value in the **bClockID** field of the Clock Source Entity descriptor (CSD). No other Clock Entity, Unit or Terminal within the AudioControl interface may have the same ID. This value must be passed in the Entity ID field (part of the **wIndex** field) of each request that is directed to the Clock Source Entity.

The **bmAttributes** field contains a Clock Type bit field (D1..0) that indicates whether the Clock Source represents an external clock (D1..0 = 0b00) or an internal clock with either fixed frequency (D1..0 = 0b01), variable frequency (D1..0 = 0b10), or programmable frequency (D1..0 = 0b11). The actual sampling frequency of the Clock Source can be retrieved through a Get Sampling Frequency request. In the programmable frequency case, the sampling frequency for this Clock Source can be set through a Set Sampling Frequency request. In addition, the Clock Source can be queried for the validity of its current sampling clock signal through a Get Clock Validity request.

Bit D2 in the **bmAttributes** field indicates whether an internal clock is free running (D2 = 0b0) or synchronized to the Start of Frame (D2 = 0b1). If D1..0 = 0b00, then D2 must also be set to 0b0.

The **bmControls** field contains a set of bit pairs, indicating which Controls are present and what their capabilities are. If a Control is present, it must be Host readable. If a certain Control is not present then the bit pair must be set to 0b00. If a Control is present but read-only, the bit pair must be set to 0b01. If a Control is also Host programmable, the bit pair must be set to 0b11. The value 0b10 is not allowed.

The **bAssocTerminal** field contains a reference to a Terminal that is associated with this Clock Source. This is useful for instance when a Clock Source's clock signal is derived from clock recovery circuitry built around a USB OUT data endpoint (represented inside the audio function by an Input Terminal). Another example would be 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.

An index to a string descriptor is provided to further describe the Clock Source Unit.

The following table presents an outline of the Clock Source descriptor.

Table 4-6: Clock Source Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	Number	Size of this descriptor, in bytes: 8
1	bDescriptorType	1	Constant	CS_INTERFACE descriptor type.
2	bDescriptorSubtype	1	Constant	CLOCK_SOURCE descriptor subtype.

Offset	Field	Size	Value	Description
3	bClockID	1	Constant	Constant uniquely identifying the Clock Source Entity within the audio function. This value is used in all requests to address this Entity.
4	bmAttributes	1	Bitmap	D1..0: Clock Type: 00: External Clock 01: Internal fixed Clock 10: Internal variable Clock 11: Internal programmable Clock D2: Clock synchronized to SOF D7..3: Reserved. Must be set to 0.
5	bmControls	1	Bitmap	D1..0: Clock Frequency Control D3..2: Clock Validity Control D7..4: Reserved. Must be set to 0.
6	bAssocTerminal	1	Constant	Terminal ID of the Terminal that is associated with this Clock Source.
7	iClockSource	1	Number	Index of a string descriptor, describing the Clock Source Entity.

4.7.2.2 Clock Selector Descriptor

The Clock Selector Entity is uniquely identified by the value in the **bClockID** field of the Clock Selector Entity descriptor (CXD). No other Clock Entity, Unit or Terminal within the AudioControl interface may have the same ID. This value must be passed in the Entity ID field (part of the **wIndex** field) of each request that is directed to the Clock Selector Entity.

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

The **bmControls** field contains a set of bit pairs, indicating which Controls are present and what their capabilities are. If a Control is present, it must be Host readable. If a certain Control is not present then the bit pair must be set to 0b00. If a Control is present but read-only, the bit pair must be set to 0b01. If a Control is also Host programmable, the bit pair must be set to 0b11. The value 0b10 is not allowed.

An index to a string descriptor is provided to further describe the Clock Selector Entity.

The following table presents an outline of the Clock Selector descriptor.

Table 4-7: Clock Selector Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	Number	Size of this descriptor, in bytes: 7+p
1	bDescriptorType	1	Constant	CS_INTERFACE descriptor type.
2	bDescriptorSubtype	1	Constant	CLOCK_SELECTOR descriptor subtype.

Offset	Field	Size	Value	Description
3	bClockID	1	Number	Constant uniquely identifying the Clock Selector Entity within the audio function. This value is used in all requests to address this Entity.
4	bNrInPins	1	Number	Number of Input Pins of this Unit: p
5	baCSourceID(1)	1	Number	ID of the Clock Entity to which the first Clock Input Pin of this Clock Selector Entity is connected.
…	…	…	…	…
5+(p-1)	baCSourceID (p)	1	Number	ID of the Clock Entity to which the last Clock Input Pin of this Clock Selector Entity is connected.
5+p	bmControls	1	Bitmap	D1..0: Clock Selector Control D7..2: Reserved. Must be set to 0.
6+p	iClockSelector	1	Index	Index of a string descriptor, describing the Clock Selector Entity.

4.7.2.3 Clock Multiplier Descriptor

The Clock Multiplier Entity is uniquely identified by the value in the **bClockID** field of the Clock Multiplier Entity descriptor (CMD). No other Clock Entity, Unit or Terminal within the AudioControl interface may have the same ID. This value must be passed in the Entity ID field (part of the **wIndex** field) of each request that is directed to the Clock Multiplier Entity.

The bCSourceID field contains the ID of the Clock Entity to which the Clock Multiplier's Clock Input Pin is connected.

The **bmControls** field contains a set of bit pairs, indicating which Controls are present and what their capabilities are. If a Control is present, it must be Host readable. If a certain Control is not present then the bit pair must be set to 0b00. If a Control is present but read-only, the bit pair must be set to 0b01. Since both Clock Numerator and Clock Denominator Controls can only be read-only, the value 0b11 is not allowed. The value 0b10 is also not allowed.

An index to a string descriptor is provided to further describe the Clock Multiplier Entity.

The following table presents an outline of the Clock Multiplier descriptor.

Table 4-8: Clock Multiplier Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	Number	Size of this descriptor, in bytes: 7
1	bDescriptorType	1	Constant	CS_INTERFACE descriptor type.
2	bDescriptorSubtype	1	Constant	CLOCK_MULTIPLIER descriptor subtype.

Offset	Field	Size	Value	Description
3	bClockID	1	Constant	Constant uniquely identifying the Clock Multiplier Entity within the audio function. This value is used in all requests to address this Entity.
4	bCSourceID	1	Number	ID of the Clock Entity to which the last Clock Input Pin of this Clock Selector Entity is connected.
5	bmControls	1	Bitmap	D1..0: Clock Numerator Control D3..2: Clock Denominator Control D7..4: Reserved. Must be set to 0.
6	iClockMultiplier	1	Index	Index of a string descriptor, describing the Clock Multiplier Entity.

4.7.2.4 Input Terminal Descriptor

The Input Terminal descriptor (ITD) 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 **bTerminalID** field. No other Unit or Terminal within the AudioControl interface may have the same ID. This value must be passed in the Entity ID field (part of the **wIndex** field) of each request that is directed to the Terminal.

The **wTerminalType** field provides pertinent information about the physical entity that the Input Terminal represents. This could be a USB OUT endpoint, an external Line In connection, a microphone, etc. A complete list of Terminal Type codes is provided in a separate document, *USB Audio Terminal Types* that is considered part of this specification.

The **bAssocTerminal** field is used to associate an Output Terminal to this Input Terminal, effectively implementing a bi-directional Terminal pair. If no association exists, the **bAssocTerminal** field must be set to zero.

The Host software can treat the associated Terminals as being physically related. In many cases, one Terminal cannot exist without the other. A typical example of such a Terminal pair is an Input Terminal, which represents the microphone, and an Output Terminal, which represents the earpiece of a headset.

The **bCSourceID** contains a constant indicating to which Clock Entity the Clock Input Pin of this Input Terminal is connected.

The **bNrChannels**, **bmChannelConfig** and **iChannelNames** fields together constitute the cluster descriptor. They characterize the cluster that leaves the Input Terminal over the single Output Pin (‘downstream’ connection). For a detailed description of the cluster descriptor, see Section 4.1, “Audio Channel Cluster Descriptor”.

The **bmControls** field contains a set of bit pairs, indicating which Controls are present and what their capabilities are. If a Control is present, it must be Host readable. If a certain Control is not present then the bit pair must be set to 0b00. If a Control is present but read-only, the bit pair must be set to 0b01. If a Control is also Host programmable, the bit pair must be set to 0b11. The value 0b10 is not allowed.

An index to a string descriptor is provided to further describe the Input Terminal.

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

Table 4-9: Input Terminal Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	Number	Size of this descriptor, in bytes: 17
1	bDescriptorType	1	Constant	CS_INTERFACE descriptor type.
2	bDescriptorSubtype	1	Constant	INPUT_TERMINAL descriptor subtype.
3	bTerminalID	1	Constant	Constant uniquely identifying the Terminal within the audio function. This value is used in all requests to address this Terminal.
4	wTerminalType	2	Constant	Constant characterizing the type of Terminal. See *USB Audio Terminal Types*.
6	bAssocTerminal	1	Constant	ID of the Output Terminal to which this Input Terminal is associated.
7	bCSourceID	1	Constant	ID of the Clock Entity to which this Input Terminal is connected.
8	bNrChannels	1	Number	Number of logical output channels in the Terminal's output audio channel cluster.
9	bmChannelConfig	4	Bitmap	Describes the spatial location of the logical channels.
13	iChannelNames	1	Index	Index of a string descriptor, describing the name of the first logical channel.
14	bmControls	2	Bitmap	D1..0: Copy Protect Control D3..2: Connector Control D5..4: Overload Control D7..6: Cluster Control D9..8: Underflow Control D11..10: Overflow Control D15..12: Reserved. Must be set to 0.
16	iTerminal	1	Index	Index of a string descriptor, describing the Input Terminal.

4.7.2.5 Output Terminal Descriptor

The Output Terminal descriptor (OTD) 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 **bTerminalID** field. No other Unit or Terminal within the AudioControl interface may have the same ID. This value must be passed in the Entity ID field (part of the **wIndex** field) of each request that is directed to the Terminal.

The **wTerminalType** field provides pertinent information about the physical entity the Output Terminal represents. This could be a USB IN endpoint, an external Line Out connection, a speaker system etc. A complete list of Terminal Type codes is provided in a separate document, *USB Audio Terminal Types* that is considered part of this specification.

The **bAssocTerminal** field is used to associate an Input Terminal to this Output Terminal, effectively implementing a bi-directional Terminal pair. If no association exists, the **bAssocTerminal** field must be set to zero.

The Host software can treat the associated Terminals as being physically related. In many cases, one Terminal cannot exist without the other. A typical example of such a Terminal pair is an Input Terminal, which represents the microphone, and an Output Terminal, which represents the earpiece of a headset.

The **bSourceID** 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 audio channel cluster.

The **bCSourceID** contains a constant indicating to which Clock Entity the Clock Input Pin of this Output Terminal is connected.

The **bmControls** field contains a set of bit pairs, indicating which Controls are present and what their capabilities are. If a Control is present, it must be Host readable. If a certain Control is not present then the bit pair must be set to 0b00. If a Control is present but read-only, the bit pair must be set to 0b01. If a Control is also Host programmable, the bit pair must be set to 0b11. The value 0b10 is not allowed.

An index to a string descriptor is provided to further describe the Output Terminal.

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

Table 4-10: Output Terminal Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	Number	Size of this descriptor, in bytes: 12
1	bDescriptorType	1	Constant	CS_INTERFACE descriptor type.
2	bDescriptorSubtype	1	Constant	OUTPUT_TERMINAL descriptor subtype.
3	bTerminalID	1	Constant	Constant uniquely identifying the Terminal within the audio function. This value is used in all requests to address this Terminal.
4	wTerminalType	2	Constant	Constant characterizing the type of Terminal. See *USB Audio Terminal Types*.
6	bAssocTerminal	1	Constant	Constant, identifying the Input Terminal to which this Output Terminal is associated.
7	bSourceID	1	Constant	ID of the Unit or Terminal to which this Terminal is connected.
8	bCSourceID	1	Constant	ID of the Clock Entity to which this Output Terminal is connected.
9	bmControls	2	Bitmap	D1..0: Copy Protect Control D3..2: Connector Control D5..4: Overload Control D7..6: Underflow Control D9..8: Overflow Control D15..10: Reserved. Must be set to 0.

Offset	Field	Size	Value	Description
11	iTerminal	1	Index	Index of a string descriptor, describing the Output Terminal.

4.7.2.6 Mixer Unit Descriptor

The Mixer Unit is uniquely identified by the value in the **bUnitID** field of the Mixer Unit descriptor (MUD). No other Unit or Terminal within the AudioControl interface may have the same ID. This value must be passed in the Entity ID field (part of the **wIndex** field) of each request that is directed to the Mixer Unit.

The **bNrInPins** field contains the number of Input Pins (*p*) of the Mixer Unit. This evidently equals the number of audio channel clusters that enter the Mixer Unit. The connectivity of the Input Pins is described via the **baSourceID()** array, containing p elements. The index *i* into the array is one-based and directly related to the Input Pin numbers. **baSourceID(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 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 audio channel clusters.

Because a Mixer Unit can 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 **bNrChannels**, **bmChannelConfig** and **iChannelNames** characterize the cluster that leaves the Mixer Unit over the single Output Pin ('downstream' connection). For a detailed description of the cluster descriptor, see Section 4.1, "Audio Channel Cluster Descriptor".

As mentioned before, every input channel can virtually be mixed into all of the output channels. If *n* is the total number of logical input channels, contained in all the audio channel clusters that are entering the Mixer Unit:

$$n = \sum_{i=1}^{\text{Number of clusters}} (\text{number of logical channels in cluster i})$$

and *m* is the number of logical output channels, then there are ($n \cdot m$) Mixer Controls in the Mixer Unit, some of which may not be programmable.

Note: ($n \cdot m$) must be limited to 256.

The Mixer Unit Descriptor reports which Controls are programmable in the **bmMixerControls** bitmap field. This bitmap must be interpreted as a two-dimensional bit array that has a row for each logical input channel and a column for each logical output channel. If a bit at position [*u*, *v*] is set to one, this means that the Mixer Unit contains a programmable Mixer Control that connects input channel *u* to output channel *v*. If bit [*u*, *v*] is set to zero, this indicates that the connection between input channel *u* and output channel *v* is non-programmable. The valid range for *u* is from one to *n*. The valid range for *v* is from one to *m*.

Each Mixer Control is assigned a unique Mixer Control Number (MCN). This number is used to address a particular Mixer Control in a Get/Set Mixer Control request. The MCN is calculated as follows:

$$MCN = (u-1).m + (v-1)$$

The following figure presents a more graphical explanation.

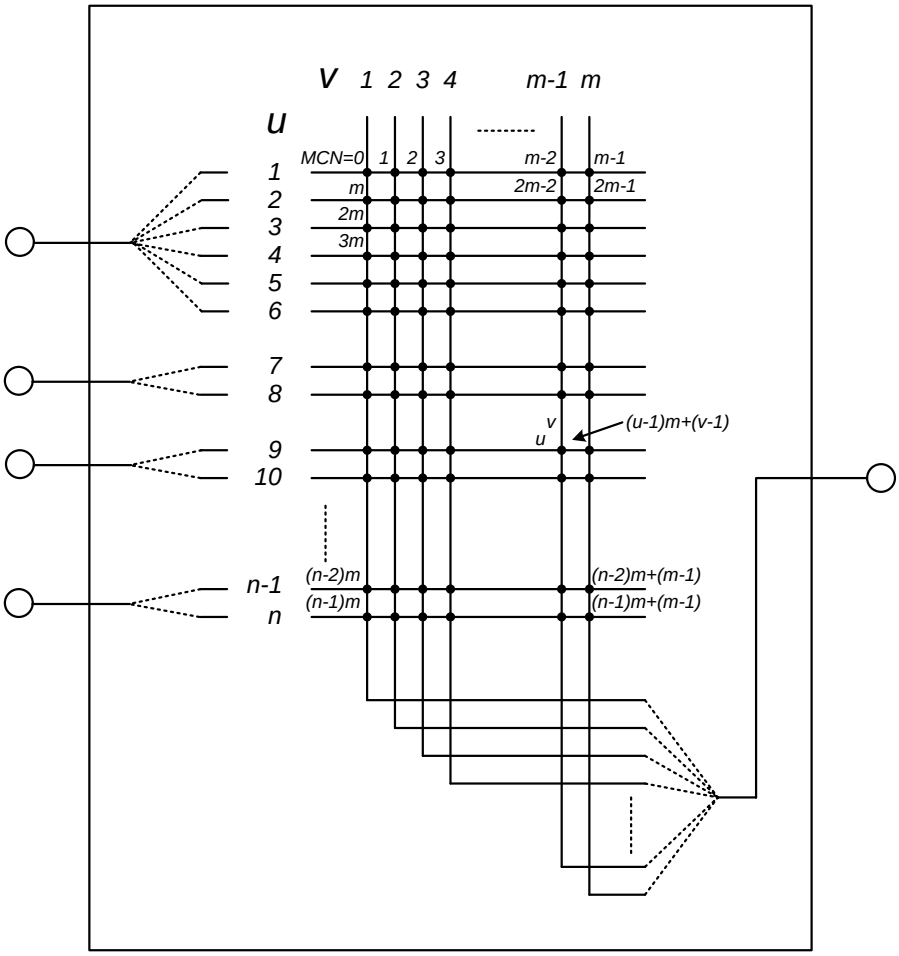

Figure 4-1: Mixer internals

The current setting of the Mixer Control (both programmable and fixed) at any position in the matrix can always be retrieved through the appropriate request. Therefore the Mixer Unit must always implement the Get request with the CUR attribute for each node in the matrix. See Section 5.2.5.5, “Mixer Unit Control Request” for further details.

The **bmMixerControls** field stores the bit array row after row where the MSb of the first byte corresponds to the connection between input channel 1 and output channel 1. If ($n \cdot m$) is not an integer multiple of 8, the bit array is padded with zeros until an integer number of bytes is occupied. The number of bytes used to store the bit array, *N*, can be calculated as follows:

```
IF ((n * m) MOD 8) <> 0 THEN
  N = ((n * m) DIV 8) + 1
ELSE
  N = ((n * m) DIV 8)
```

The **bmControls** field contains a set of bit pairs, indicating which Controls are present and what their capabilities are. If a Control is present, it must be Host readable. If a certain Control is not present then the bit pair must be set to 0b00. If a Control is present but read-only, the bit pair must be set to 0b01. If a Control is also Host programmable, the bit pair must be set to 0b11. The value 0b10 is not allowed.

An index to a string descriptor is provided to further describe the Mixer Unit.

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

Table 4-11: Mixer Unit Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	Number	Size of this descriptor, in bytes: 13+p+N
1	bDescriptorType	1	Constant	CS_INTERFACE descriptor type.
2	bDescriptorSubtype	1	Constant	MIXER_UNIT descriptor subtype.
3	bUnitID	1	Number	Constant uniquely identifying the Unit within the audio function. This value is used in all requests to address this Unit.
4	bNrInPins	1	Number	Number of Input Pins of this Unit: p
5	baSourceID(1)	1	Number	ID of the Unit or Terminal to which the first Input Pin of this Mixer Unit is connected.
…	…	…	…	…
5+(p-1)	baSourceID (p)	1	Number	ID of the Unit or Terminal to which the last Input Pin of this Mixer Unit is connected.
5+p	bNrChannels	1	Number	Number of logical output channels in the Mixer's output audio channel cluster.
6+p	bmChannelConfig	4	Bitmap	Describes the spatial location of the logical channels.
10+p	iChannelNames	1	Index	Index of a string descriptor, describing the name of the first logical channel.
11+p	bmMixerControls	N	Number	Bit map indicating which Mixer Controls are programmable.
11+p+N	bmControls	1	Bitmap	D1..0: Cluster Control D3..2: Underflow Control D5..4: Overflow Control D7..6: Reserved. Must be set to 0.
12+p+N	iMixer	1	Index	Index of a string descriptor, describing the Mixer Unit.

4.7.2.7 Selector Unit Descriptor

The Selector Unit is uniquely identified by the value in the **bUnitID** field of the Selector Unit descriptor (SUD). No other Unit or Terminal within the AudioControl interface may have the same ID. This value must be passed in the Entity ID field (part of the **wIndex** field) of each request that is directed to the Selector Unit.

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

The cluster descriptors, describing the logical channels that enter the Selector Unit are not repeated here. In order for a Selector Unit to be legally connected, **all** of the audio channel clusters that enter the Selector Unit **must** have exactly the same audio channel descriptor content. Therefore all channel clusters will have the same number of channels and the same bmChannelConfig settings.

The **bmControls** field contains a set of bit pairs, indicating which Controls are present and what their capabilities are. If a Control is present, it must be Host readable. If a certain Control is not present then the bit pair must be set to 0b00. If a Control is present but read-only, the bit pair must be set to 0b01. If a Control is also Host programmable, the bit pair must be set to 0b11. The value 0b10 is not allowed.

An index to a string descriptor is provided to further describe the Selector Unit.

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

Table 4-12: Selector Unit Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	Number	Size of this descriptor, in bytes: 7+p
1	bDescriptorType	1	Constant	CS_INTERFACE descriptor type.
2	bDescriptorSubtype	1	Constant	SELECTOR_UNIT descriptor subtype.
3	bUnitID	1	Number	Constant uniquely identifying the Unit within the audio function. This value is used in all requests to address this Unit.
4	bNrInPins	1	Number	Number of Input Pins of this Unit: p
5	baSourceID(1)	1	Number	ID of the Unit or Terminal to which the first Input Pin of this Selector Unit is connected.
…	…	…	…	…
5+(p-1)	baSourceID (p)	1	Number	ID of the Unit or Terminal to which the last Input Pin of this Selector Unit is connected.
5+p	bmControls	1	Bitmap	D1..0: Selector Control D7..2: Reserved. Must be set to 0.
6+p	iSelector	1	Index	Index of a string descriptor, describing the Selector Unit.

4.7.2.8 Feature Unit Descriptor

The Feature Unit is uniquely identified by the value in the **bUnitID** field of the Feature Unit descriptor (FUD). No other Unit or Terminal within the AudioControl interface may have the same ID. This value must be passed in the Entity ID field (part of the **wIndex** field) of each request that is directed to the Feature Unit.

The **bSourceID** 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 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 audio channel cluster.

bmaControls() is a (ch+1)-element array of 4-byte bitmaps, each containing a set of bit pairs. If a Control is present, it must be Host readable. If a certain Control is not present then the bit pair must be set to 0b00. If a Control is present but read-only, the bit pair must be set to 0b01. If a Control is also Host programmable, the bit pair must be set to 0b11. The value 0b10 is not allowed.

An index to a string descriptor is provided to further describe the Feature Unit.

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

Table 4-13: Feature Unit Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	Number	Size of this descriptor, in bytes: 6+(ch+1)*4
1	bDescriptorType	1	Constant	CS_INTERFACE descriptor type.
2	bDescriptorSubtype	1	Constant	FEATURE_UNIT descriptor subtype.
3	bUnitID	1	Number	Constant uniquely identifying the Unit within the audio function. This value is used in all requests to address this Unit.
4	bSourceID	1	Constant	ID of the Unit or Terminal to which this Feature Unit is connected.
5	bmaControls(0)	4	Bitmap	The Controls bitmap for master channel 0: D1..0: Mute Control D3..2: Volume Control D5..4: Bass Control D7..6: Mid Control D9..8: Treble Control D11..10: Graphic Equalizer Control D13..12: Automatic Gain Control D15..14: Delay Control D17..16: Bass Boost Control D19..18: Loudness Control D21..20: Input Gain Control D23..22: Input Gain Pad Control D25..24: Phase Inverter Control D27..26: Underflow Control D29..28: Overfow Control D31..30: Reserved. Must be set to 0.
5+(1*4)	bmaControls(1)	4	Bitmap	The Controls bitmap for logical channel 1.
…	…	…	…	…
5+(ch*4)	bmaControls(ch)	4	Bitmap	The Controls bitmap for logical channel ch.
5+(ch+1)*4	iFeature	1	Index	Index of a string descriptor, describing this Feature Unit.

4.7.2.9 Sampling Rate Converter Descriptor

The Sampling Rate Converter Unit descriptor (RUD) 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 **bUnitID** field. No other Unit or Terminal within the AudioControl interface may have the same ID. This value must be passed in the Entity ID field (part of the **wIndex** field) of each request that is directed to the Feature Unit.

The **bSourceID** 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 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 audio channel cluster.

The **bCSourceInID** contains a constant indicating to which Clock Entity the Clock Input Pin associated with the audio Input Pin is connected.

The **bCSourceOutID** contains a constant indicating to which Clock Entity the Clock Input Pin associated with the audio Output Pin is connected.

An index to a string descriptor is provided to further describe the SRC Unit.

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

Table 4-14: Sampling Rate Converter Unit Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	Number	Size of this descriptor, in bytes: 8
1	bDescriptorType	1	Constant	CS_INTERFACE descriptor type.
2	bDescriptorSubtype	1	Constant	SAMPLE_RATE_CONVERTER descriptor subtype.
3	bUnitID	1	Constant	Constant uniquely identifying the Unit within the audio function. This value is used in all requests to address this Unit.
4	bSourceID	1	Constant	ID of the Unit or Terminal to which this SRC Unit is connected.
5	bCSourceInID	1	Constant	ID of the Clock Entity to which this SRC Unit input section is connected.
6	bCSourceOutID	1	Constant	ID of the Clock Entity to which this SRC Unit output section is connected.
7	iSRC	1	Index	Index of a string descriptor, describing the SRC Unit.

4.7.2.10 Effect Unit Descriptor

The Effect Unit is uniquely identified by the value in the **bUnitID** field of the Effect Unit descriptor (EUD). No other Unit or Terminal within the AudioControl interface may have the same ID. This value must be passed in the Entity ID field (part of the **wIndex** field) of each request 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.11, "Effect Unit Effect Types."

The **bSourceID** 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 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 audio channel cluster.

bmaControls() is a (ch+1)-element array of 4-byte bitmaps, each containing a set of bit pairs. If a Control is present, it must be Host readable. If a certain Control is not present then the bit pair must be set to 0b00. If a Control is present but read-only, the bit pair must be set to 0b01. If a Control is also Host programmable, the bit pair must be set to 0b11. The value 0b10 is not allowed.

An index to a string descriptor is provided to further describe the Effect Unit.

The previous fields are common to all Effect Units. However, depending on the value in the **wEffectType** field, a effect-specific part is added to the descriptor. The following paragraphs describe these effect-specific parts.

The following table outlines the common part of the Effect Unit descriptor.

Table 4-15: Common Part of the Effect Unit Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	Number	Size of this descriptor, in bytes: 16+(ch*4)
1	bDescriptorType	1	Constant	CS_INTERFACE descriptor type.
2	bDescriptorSubtype	1	Constant	EFFECT_UNIT descriptor subtype.
3	bUnitID	1	Number	Constant uniquely identifying the Unit within the audio function. This value is used in all requests to address this Unit.
4	wEffectType	2	Constant	Constant identifying the type of effect this Unit is performing.
6	bSourceID	1	Constant	ID of the Unit or Terminal to which this Effect Unit is connected.
7	bmaControls(0)	4	Bitmap	The Controls bitmap for master channel 0: D31..0: Effect-specific allocation.
11	bmaControls(1)	4	Bitmap	The Controls bitmap for master channel 1: D31..0: Effect-specific allocation.
…	…	…	…	…
11+(ch*4)	bmaControls(ch)	4	Bitmap	The Controls bitmap for master channel ch: D31..0: Effect-specific allocation.
15+(ch*4)	iEffects	1	Index	Index of a string descriptor, describing this Effect Unit.

4.7.2.10.1 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.11, “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 4-16: Parametric Equalizer Section Effect Unit Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	Number	Size of this descriptor, in bytes: 16+(ch*4)
1	bDescriptorType	1	Constant	CS_INTERFACE descriptor type.
2	bDescriptorSubtype	1	Constant	EFFECT_UNIT descriptor subtype.
3	bUnitID	1	Number	Constant uniquely identifying the Unit within the audio function. This value is used in all requests to address this Unit.
4	wEffectType	2	Constant	PARAM_EQ_SECTION_EFFECT effect type.
6	bSourceID	1	Constant	ID of the Unit or Terminal to which this Effect Unit is connected.
7	bmaControls(0)	4	Bitmap	The Controls bitmap for master channel 0: D1..0: Enable Control D3..2: Center Frequency Control D5..4: Q Factor Control D7..6: Gain Control D9..8: Underflow Control D11..10: Overflow Control D31..12: Reserved. Must be set to 0
11	bmaControls(1)	4	Bitmap	The Controls bitmap for logical channel 1.
…	…	…	…	…
11+(ch*4)	bmaControls(ch)	4	Bitmap	The Controls bitmap for logical channel ch.
15+(ch*4)	iEffects	1	Index	Index of a string descriptor, describing this Effect Unit.

4.7.2.10.2 Reverberation Effect Unit Descriptor

The **wEffectType** field of the common Effect Unit descriptor contains the value REVERBERATION_EFFECT. (See Appendix A.11, “Effect Unit Effect Types”)

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

Table 4-17: Reverberation Effect Unit Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	Number	Size of this descriptor, in bytes: 16+(ch*4)
1	bDescriptorType	1	Constant	CS_INTERFACE descriptor type.
2	bDescriptorSubtype	1	Constant	EFFECT_UNIT descriptor subtype.
3	bUnitID	1	Number	Constant uniquely identifying the Unit within the audio function. This value is used in all requests to address this Unit.
4	wEffectType	2	Constant	REVERBERATION_EFFECT effect type.
6	bSourceID	1	Constant	ID of the Unit or Terminal to which this Effect Unit is connected.
7	bmaControls(0)	4	Bitmap	The Controls bitmap for master channel 0: D1..0: Enable Control D3..2: Type Control D5..4: Level Control D7..6: Time Control D9..8: Delay Feedback Control D11..10: Pre-Delay Control D13..12: Density Control D15..14: Hi-Freq Roll-Off Control D17..16: Underflow Control D19..18: Overflow Control D31..20: Reserved. Must be set to 0
11	bmaControls(1)	4	Bitmap	The Controls bitmap for logical channel 1.
…	…	…	…	…
11+(ch*4)	bmaControls(ch)	4	Bitmap	The Controls bitmap for logical channel ch.
15+(ch*4)	iEffects	1	Index	Index of a string descriptor, describing this Effect Unit.

4.7.2.10.3 Modulation Delay Effect Unit Descriptor

The **wEffectType** field of the common Effect Unit descriptor contains the value MOD_DELAY_EFFECT. (See Appendix A.11, “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 4-18: Modulation Delay Effect Unit Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	Number	Size of this descriptor, in bytes: 16+(ch*4)
1	bDescriptorType	1	Constant	CS_INTERFACE descriptor type.

Offset	Field	Size	Value	Description
2	bDescriptorSubtype	1	Constant	EFFECT_UNIT descriptor subtype.
3	bUnitID	1	Number	Constant uniquely identifying the Unit within the audio function. This value is used in all requests to address this Unit.
4	wEffectType	2	Constant	MOD_DELAY_EFFECT effect type.
6	bSourceID	1	Constant	ID of the Unit or Terminal to which this Effect Unit is connected.
7	bmaControls(0)	4	Bitmap	The Controls bitmap for master channel 0: D1..0: Enable Control D3..2: Balance Control D5..4: Rate Control D7..6: Depth Control D9..8: Time Control D11..10: Feedback Level Control D13..12: Underflow Control D15..14: Overflow Control D31..16: Reserved. Must be set to 0
11	bmaControls(1)	4	Bitmap	The Controls bitmap for logical channel 1.
…	…	…	…	…
11+(ch*4)	bmaControls(ch)	4	Bitmap	The Controls bitmap for logical channel ch.
15+(ch*4)	iEffects	1	Index	Index of a string descriptor, describing this Effect Unit.

4.7.2.10.4 Dynamic Range Compressor Effect Unit Descriptor

The **wEffectType** field of the common Effect Unit descriptor contains the value DYN_RANGE_COMP_EFFECT. (See Appendix A.11, “Effect Unit Effect Types”)

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

Table 4-19: Dynamic Range Compressor Effect Unit Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	Number	Size of this descriptor, in bytes: 16+(ch*4)
1	bDescriptorType	1	Constant	CS_INTERFACE descriptor type.
2	bDescriptorSubtype	1	Constant	EFFECT_UNIT descriptor subtype.
3	bUnitID	1	Number	Constant uniquely identifying the Unit within the audio function. This value is used in all requests to address this Unit.

Offset	Field	Size	Value	Description
4	wEffectType	2	Constant	DYN_RANGE_COMP_EFFECT effect type.
6	bSourceID	1	Constant	ID of the Unit or Terminal to which this Effect Unit is connected.
7	bmaControls(0)	4	Bitmap	The Controls bitmap for master channel 0: D1..0: Enable Control D3..2: Compression Ratio Control D5..4: MaxAmpl Control D7..6: Threshold Control D9..8: Attack Time Control D11..10: Release Time Control D13..12: Underflow Control D15..14: Overflow Control D31..16: Reserved. Must be set to 0
11	bmaControls(1)	4	Bitmap	The Controls bitmap for logical channel 1.
…	…	…	…	…
11+(ch*4)	bmaControls(ch)	4	Bitmap	The Controls bitmap for logical channel ch.
15+(ch*4)	iEffects	1	Index	Index of a string descriptor, describing this Effect Unit.

4.7.2.11 Processing Unit Descriptor

The Processing Unit is uniquely identified by the value in the **bUnitID** field of the Processing Unit descriptor (PUD). No other Unit or Terminal within the AudioControl interface may have the same ID. This value must be passed in the Entity ID field (part of the **wIndex** field) of each request 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.12, “Processing Unit Process Types.”

The **bNrInPins** field contains the number of Input Pins (*p*) of the Processing Unit. The connectivity of the Input Pins is described via the **baSourceID()** array that contains *p* elements. The index *i* into the array is one-based and directly related to the Input Pin numbers. **baSourceID(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 audio channel clusters.

Because a Processing Unit can freely redefine the spatial locations of the logical output channels, contained in its output cluster, there is a need for an output cluster descriptor. The **bNrChannels**, **bmChannelConfig,** and **iChannelNames** fields characterize the cluster that leaves the Processing Unit over the single Output Pin (‘downstream’ connection). For a detailed description of the cluster descriptor, see Section 4.1, “Audio Channel Cluster Descriptor”.

The **bmControls** field is a 2-byte bitmap containing a set of bit pairs. If a Control is present, it must be Host readable. If a certain Control is not present then the bit pair must be set to 0b00. If a Control is present but read-only, the bit pair must be set to 0b01. If a Control is also Host programmable, the bit pair must be set to 0b11. The value 0b10 is not allowed.

In general, all Controls are optional. However, some Processing Types may define certain Controls as mandatory.

The meaning of the bits in the **bmControls** field is qualified by the **wProcessType** field. However, bits D1..0 always represent the Enable Control for all Processing Unit Types. The Enable Control is used to bypass the entire functionality of the Processing Unit. When it is set to off, the behavior is as follows:

- In case of a single Input Pin, logical channels entering the Unit are passed unaltered for those channels that are also present in the output cluster. Logical channels not available in the output cluster are absorbed by the Processing Unit. Logical channels present in the output cluster but unavailable in the input cluster are muted.
- In case of multiple Input Pins, corresponding logical input channels are equally mixed together before being passed to the output.

An index to a string descriptor is provided to further describe the Processing Unit.

The previous fields are common to all Processing Units. However, depending on the value in the **wProcessType** field, a process-specific part is added to the descriptor. The following paragraphs describe these process-specific parts.

The following table outlines the common part of the Processing Unit descriptor.

Table 4-20: Common Part of the Processing Unit Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	Number	Size of this descriptor, in bytes: 17+p+x
1	bDescriptorType	1	Constant	CS_INTERFACE descriptor type.
2	bDescriptorSubtype	1	Constant	PROCESSING_UNIT descriptor subtype.
3	bUnitID	1	Number	Constant uniquely identifying the Unit within the audio function. This value is used in all requests to address this Unit.
4	wProcessType	2	Constant	Constant identifying the type of processing this Unit is performing.
6	bNrInPins	1	Number	Number of Input Pins of this Unit: p
7	baSourceID(1)	1	Number	ID of the Unit or Terminal to which the first Input Pin of this Processing Unit is connected.
…	…	…	…	…
7+(p-1)	baSourceID (p)	1	Number	ID of the Unit or Terminal to which the last Input Pin of this Processing Unit is connected.
7+p	bNrChannels	1	Number	Number of logical output channels in the audio channel cluster of the Processing Unit.
8+p	bmChannelConfig	4	Bitmap	Describes the spatial location of the logical channels in the audio channel cluster of the Processing Unit.

Offset	Field	Size	Value	Description
12+p	iChannelNames	1	Index	Index of a string descriptor that describes the name of the first logical channel in the audio channel cluster of the Processing Unit.
13+p	bmControls	2	Bitmap	D1..0: Enable Control D15..2: Process-specific allocation.
15+p	iProcessing	1	Index	Index of a string descriptor, describing this Processing Unit.
16+p	Process-specific	x	NA	A process-specific descriptor is appended to the common descriptor. See the following paragraphs.

4.7.2.11.1 Up/Down-mix Processing Unit Descriptor

The **wProcessType** field of the common Processing Unit descriptor contains the value UP/DOWNMIX_PROCESS. (See Appendix A.12, "Processing Unit Process Types")

The Up/Down-mix Processing Unit has a single Input Pin. Therefore, the **bNrInputs** field must contain the value 1. The **bNrChannels**, **bmChannelConfig,** and **iChannelNames** fields together constitute the output cluster descriptor of the Up/Down-mix Processing Unit. It describes which logical channels are physically present at the output of the Processing Unit. Depending upon the selected operating mode, one or more channels may be unused.

The **bmControls** field contains a set of bit pairs, indicating which Controls are present and what their capabilities are. If a Control is present, it must be Host readable. If a certain Control is not present then the bit pair must be set to 0b00. If a Control is present but read-only, the bit pair must be set to 0b01. If a Control is also Host programmable, the bit pair must be set to 0b11. The value 0b10 is not allowed.

Bits D1..0 of the **bmControls** field represent the Enable Control. The Mode Select Control (D3..2) is used to change the behavior of the Processing Unit by selecting different modes of operation.

The process-specific descriptor of the Up/Down-mix Processing Unit describes the supported modes of operation of the Processing Unit. Selecting a mode of operation is done by issuing the Set Mode Request. The number of supported modes (*m*) is contained in the **bNrModes** field. This field is followed by an array of mode fields, **daModes()**. The index *i* into this array is one-based and directly related to the number of the mode described by entry **daModes(i)**. It is the value *i* that must be used as a parameter for the Set Mode request to select the mode *i*.

The bit allocations in the **daModes()** fields are very similar to those of the **bmChannelConfig** field in a cluster descriptor (see Section 4.1, "Audio Channel Cluster Descriptor"). i.e. a bit set in the **daModes(i)** field indicates that for mode *i*, the Up/Down-mix Processing Unit produces meaningful audio data for the logical channel that is associated with the position of the set bit. Logical channels that are present in the output cluster but are not used in a certain mode are considered to be inactive and at most produce silence in that mode.

Each **daModes(i)** field can only contain set bits for those logical channels that are present in the output channel cluster. In other words, all **daModes()** fields can only contain a subset of the **bmChannelConfig** field of the cluster descriptor of the Unit. Furthermore, logical channels that have a non-predefined spatial position cannot be marked as active in the **daModes()** fields. It is therefore assumed by default that they are active.

The following table outlines the combination of the common and process-specific Up/Down-mix Processing Unit descriptors.

Table 4-21: Up/Down-mix Processing Unit Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	Number	Size of this descriptor, in bytes: 18+4*m
1	bDescriptorType	1	Constant	CS_INTERFACE descriptor type.
2	bDescriptorSubtype	1	Constant	PROCESSING_UNIT descriptor subtype.
3	bUnitID	1	Number	Constant uniquely identifying the Unit within the audio function. This value is used in all requests to address this Unit.
4	wProcessType	2	Constant	UP/DOWNMIX_PROCESS process type.
6	bNrInPins	1	Number	Number of Input Pins of this Unit: 1
7	bSourceID	1	Number	ID of the Unit or Terminal to which the Input Pin of this Processing Unit is connected.
8	bNrChannels	1	Number	Number of logical output channels in the output channel cluster of the Processing Unit.
9	bmChannelConfig	4	Bitmap	Describes the spatial location of the logical channels in the output channel cluster of the Processing Unit.
13	iChannelNames	1	Index	Index of a string descriptor, describing the name of the first logical channel in the Processing Unit's output channel cluster.
14	bmControls	2	Bitmap	D1..0: Enable Control D3..2: Mode Select Control D5..4: Cluster Control D7..6: Underflow Control D9..8: Overflow Control D15..10: Reserved. Must be set to 0
16	iProcessing	1	Index	Index of a string descriptor, describing this Processing Unit.
17	bNrModes	1	Number	Number of modes, supported by this Processing Unit: m
18	daModes(1)	4	Bitmap	Describes the active logical channels in mode 1.
…	…	…	…	…
18+4*(m-1)	daModes(m)	4	Bitmap	Describes active the logical channels in mode m.

4.7.2.11.2 Dolby Prologic Processing Unit Descriptor

The **wProcessType** field of the common Processing Unit descriptor contains the value DOLBY_PROLOGIC_PROCESS. (See Appendix A.12, "Processing Unit Process Types")

The Dolby Prologic Processing Unit has a single Input Pin. Therefore, the **bNrInputs** field must contain the value 1. The **bNrChannels**, **bmChannelConfig**, and **iChannelNames** fields together constitute the output cluster descriptor of the Dolby Prologic Processing Unit. It describes which logical channels are physically present at the output of the Processing Unit. Depending upon the selected operating mode, one or more channels may be unused.

Bits D1..0 of the **bmControls** field represent the Enable Control. The Mode Select Control (D3..2) is used to change the behavior of the Processing Unit by selecting different modes of operation.

Although the input cluster may contain a number of logical channels, the Dolby Prologic Processing Unit only uses Left and Right logical input channels as input for the decoding process. Obviously, these two logical channels must be present in the input cluster for the Unit to operate properly. All other logical channels are discarded.

The output cluster may contain logical channels other than Left, Right, Center, and/or Surround (these must be present) to facilitate connectivity within the audio function. Channels that are present in the output cluster but do not participate in the chosen mode of operation must be muted.

The process-specific descriptor of the Dolby Prologic Processing Unit describes the supported modes of operation of the Processing Unit. The number of supported modes (*m*) is contained in the **bNrModes** field. This field is followed by an array of mode fields, **daModes()**. The bit allocations in the **daModes()** fields are very similar to those of the **bmChannelConfig** field in a cluster descriptor (see Section 4.1, "Audio Channel Cluster Descriptor"), i.e., a bit set in the **daModes(i)** field indicates that for mode *i*, the Dolby Prologic Processing Unit produces meaningful audio data for the logical channel that is associated with the position of the set bit.

The Dolby Prologic Processing Unit currently supports the following three different modes:

- Left, Right, Center channel decoding **daModes()** = 0x00000007
- Left, Right, Surround channel decoding **daModes()** = 0x00000103
- Left, Right, Center, Surround decoding **daModes()** = 0x00000107

The **bmChannelConfig** field of the cluster descriptor of the Unit must at least contain the union of all bits set for all the supported modes.

The following table outlines the combination of the common and process-specific Dolby Prologic Processing Unit descriptors. It is identical to the Up/Down-mix Processing Unit descriptor except for some field values. It is repeated here for clarity.

Table 4-22: Dolby Prologic Processing Unit Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	Number	Size of this descriptor, in bytes: 18+4*m
1	bDescriptorType	1	Constant	CS_INTERFACE descriptor type.
2	bDescriptorSubtype	1	Constant	PROCESSING_UNIT descriptor subtype.
3	bUnitID	1	Number	Constant uniquely identifying the Unit within the audio function. This value is used in all requests to address this Unit.
4	wProcessType	2	Constant	DOLBY_PROLOGIC_PROCESS process type.

Offset	Field	Size	Value	Description
6	bNrInPins	1	Number	Number of Input Pins of this Unit: 1
7	bSourceID	1	Number	ID of the Unit or Terminal to which the Input Pin of this Processing Unit is connected.
8	bNrChannels	1	Number	Number of logical output channels in the output channel cluster of the Processing Unit.
9	bmChannelConfig	4	Bitmap	Describes the spatial location of the logical channels in the Processing Unit's output channel cluster. At least Left, Right, Center and/or Surround must be set.
13	iChannelNames	1	Index	Index of a string descriptor, describing the name of the first logical channel in the Processing Unit's output channel cluster.
14	bmControls	2	Bitmap	D1..0: Enable Control D3..2: Mode Select Control D5..4: Cluster Control D7..6: Underflow Control D9..8: Overflow Control D15..10: Reserved. Must be set to 0
16	iProcessing	1	Index	Index of a string descriptor, describing this Processing Unit.
17	bNrModes	1	Number	Number of modes, supported by this Processing Unit: m A maximum of 3 different modes is possible.
18	daModes(1)	4	Bitmap	Describes the active logical channels in mode 1.
…	…	…	…	…
18+4*(m-1)	daModes(m)	4	Bitmap	Describes active the logical channels in mode m.

4.7.2.11.3 Stereo Extender Processing Unit Descriptor

The **wProcessType** field of the common Processing Unit descriptor contains the value STEREO_EXTENDER_PROCESS. (See Appendix A.12, "Processing Unit Process Types")

The Stereo Extender Processing Unit has a single Input Pin. Therefore, the **bNrInputs** field must contain the value 1. The **bNrChannels**, **bmChannelConfig** and **iChannelNames** fields together constitute the output cluster descriptor of the Stereo Extender Processing Unit. It describes which logical channels are physically present at the output of the Processing Unit.

Bits D1..0 of the **bmControls** field represent the Enable Control. Bits D3..2 indicate the availability of the Width Control.

Although the input cluster may contain a number of logical channels, the Stereo Extender Processing Unit only uses Left and Right logical input channels as input for the extension process. Obviously, these two logical channels must be present in the input cluster for the Unit to operate properly. All other logical channels are ignored by the process.

The output cluster may contain logical channels other than Left and Right (these must be present) to facilitate connectivity within the audio function. Channels that are present in the output cluster but not in the input cluster must be muted. Channels other than Left and Right that are present in both input and output cluster can be passed unaltered from input to output. Channels only present in the input cluster are absorbed by the Processing Unit.

There is no process-specific descriptor for the Stereo Extender Processing Unit.

The following table outlines the Stereo Extender Processing Unit descriptor. It is identical to the common Processing Unit descriptor, except for some field values. It is repeated here for clarity.

Table 4-23: Stereo Extender Processing Unit Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	Number	Size of this descriptor, in bytes: 17
1	bDescriptorType	1	Constant	CS_INTERFACE descriptor type.
2	bDescriptorSubtype	1	Constant	PROCESSING_UNIT descriptor subtype.
3	bUnitID	1	Number	Constant uniquely identifying the Unit within the audio function. This value is used in all requests to address this Unit.
4	wProcessType	2	Constant	STEREO_EXTENDER_PROCESS process type.
6	bNrInPins	1	Number	Number of Input Pins of this Unit: 1
7	bSourceID	1	Number	ID of the Unit or Terminal to which the Input Pin of this Processing Unit is connected.
8	bNrChannels	1	Number	Number of logical output channels in the Processing Unit’s output channel cluster.
9	bmChannelConfig	4	Bitmap	Describes the spatial location of the logical channels in the output channel cluster of the Processing Unit. At least Left and Right must be set.
13	iChannelNames	1	Index	Index of a string descriptor, describing the name of the first logical channel in the Processing Unit’s output channel cluster.
14	bmControls	2	Bitmap	D1..0: Enable Control D3..2: Width Control D5..4: Cluster Control D7..6: Underflow Control D9..8: Overflow Control D15..10: Reserved. Must be set to 0

Offset	Field	Size	Value	Description
16	iProcessing	1	Index	Index of a string descriptor, describing this Processing Unit.

4.7.2.12 Extension Unit Descriptor

The Extension Unit is uniquely identified by the value in the **bUnitID** field of the Extension Unit descriptor (XUD). No other Unit or Terminal within the AudioControl interface may have the same ID. This value must be passed in the Entity ID field (part of the **wIndex** field) of each request that is directed to the Extension Unit.

The Extension Unit descriptor provides minimal information about the Extension Unit for a generic driver at least to notice the presence of vendor-specific components within the audio function. The **wExtensionCode** field may contain a vendor-specific code that further identifies the Extension Unit. If it is not used, it should be set to zero.

The **bNrInPins** field contains the number of Input Pins (*p*) of the Extension Unit. The connectivity of the Input Pins is described via the **baSourceID()** array that contains *p* elements. The index *i* into the array is one-based and directly related to the Input Pin numbers. **baSourceID(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 audio channel clusters.

Because an Extension Unit can freely redefine the spatial locations of the logical output channels that are contained in its output cluster, there is a need for an output cluster descriptor. The **bNrChannels**, **bmChannelConfig**, and **iChannelNames** fields characterize the cluster that leaves the Extension Unit over its single Output Pin ('downstream' connection). For a detailed description of the cluster descriptor, see Section 4.1, "Audio Channel Cluster Descriptor".

The **bmControls** field is a 1-byte bitmap containing a set of bit pairs. If a Control is present, it must be Host readable. If a certain Control is not present then the bit pair must be set to 0b00. If a Control is present but read-only, the bit pair must be set to 0b01. If a Control is also Host programmable, the bit pair must be set to 0b11. The value 0b10 is not allowed.

The Enable Control must be supported by every Extension Unit and must default to off. Therefore, the bit pair D1..0 in the **bmControls** field indicates whether the Enable Control is read-only (D1..0 = 0b01) or Host programmable (D1..0 = 0b11). All other values are not allowed.

The Enable Control is used to bypass the entire functionality of the Extension Unit. This Control is mandatory for it allows a generic driver to operate the audio function without further knowledge of the internals of the Extension Unit. (Of course, the additional functionality provided by the Extension Unit is not available in this case because it is bypassed).

Default behavior is assumed when the Enable Control is set to off.

- In the case of a single Input Pin, logical channels that enter the Extension Unit are passed unaltered for those channels that are also present in the output cluster. Logical channels not available in the output cluster are absorbed (muted) by the Extension Unit. Logical channels present in the output cluster but unavailable in the input cluster produce silence.
- In case of multiple Input Pins, corresponding logical input channels that are also present in the output cluster are equally mixed together before being passed to the output channels. Logical channels not present in all input clusters are assumed to be present but muted in those clusters before mixing. Logical channels not available in the output cluster are absorbed (muted) by the Extension Unit. Logical channels present in the output cluster but unavailable in any of the input clusters produce silence.

An index to a string descriptor is provided to further describe the Extension Unit.

The following table outlines the Extension Unit descriptor.

Table 4-24: Extension Unit Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	Number	Size of this descriptor, in bytes: 16+p
1	bDescriptorType	1	Constant	CS_INTERFACE descriptor type.
2	bDescriptorSubtype	1	Constant	EXTENSION_UNIT descriptor subtype.
3	bUnitID	1	Number	Constant uniquely identifying the Unit within the audio function. This value is used in all requests to address this Unit.
4	wExtensionCode	2	Constant	Vendor-specific code identifying the Extension Unit.
6	bNrInPins	1	Number	Number of Input Pins of this Unit: p
7	baSourceID(1)	1	Number	ID of the Unit or Terminal to which the first Input Pin of this Extension Unit is connected.
…	…	…	…	…
8+(p-1)	baSourceID (p)	1	Number	ID of the Unit or Terminal to which the last Input Pin of this Extension Unit is connected.
8+p	bNrChannels	1	Number	Number of logical output channels in the audio channel cluster of the Extension Unit.
9+p	bmChannelConfig	4	Bitmap	Describes the spatial location of the logical channels in the audio channel cluster of the Extension Unit.
13+p	iChannelNames	1	Index	Index of a string descriptor, describing the name of the first logical channel in the audio channel cluster of the Extension Unit.
14+p	bmControls	1	Bitmap	D1..0: Enable Control D3..2: Cluster Control D5..4: Underflow Control D7..6: Overflow Control
15+p	iExtension	1	Index	Index of a string descriptor, describing this Extension Unit.

4.8 AudioControl Endpoint Descriptors

The following sections describe all possible endpoint-related descriptors for the AudioControl interface.

4.8.1 AC Control Endpoint Descriptors

4.8.1.1 Standard AC Control Endpoint Descriptor

Because endpoint 0 is used as the AudioControl control endpoint, there is no dedicated standard control endpoint descriptor.

4.8.1.2 Class-Specific AC Control Endpoint Descriptor

There is no dedicated class-specific control endpoint descriptor.

4.8.2 AC Interrupt Endpoint Descriptors

4.8.2.1 Standard AC Interrupt Endpoint Descriptor

The interrupt endpoint descriptor is identical to the standard endpoint descriptor defined in Section 9.6.6, "Endpoint," of the *USB Specification* and further expanded as defined in the *Universal Serial Bus Class Specification*. Its fields are set to reflect the interrupt type of the endpoint. This endpoint is optional.

The following table outlines the standard AC Interrupt Endpoint descriptor.

Table 4-25: Standard AC 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
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, reset to zero D3..0: The endpoint number, determined by the designer.
3	bmAttributes	1	Bit Map	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 information.
6	bInterval	1	Number	Interval for polling the Interrupt endpoint.

4.8.2.2 Class-Specific AC Interrupt Endpoint Descriptor

There is no class-specific AudioControl interrupt endpoint descriptor.

4.9 AudioStreaming Interface Descriptors

The AudioStreaming (AS) interface descriptors contain all relevant information to characterize the AudioStreaming interface in full.

4.9.1 Standard AS Interface Descriptor

The standard AS interface descriptor is identical to the standard interface descriptor defined in Section 9.6.3, "Interface," of the *USB Specification*, except that some fields now have dedicated values.

Table 4-26: Standard AS 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.
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). Must be either 0 (no data endpoint), 1 (data endpoint) or 2 (data and explicit feedback endpoint).
5	bInterfaceClass	1	Class	AUDIO Audio Interface Class code (assigned by the USB). See Appendix A.4, "Audio Interface Class Code"
6	bInterfaceSubClass	1	Subclass	AUDIO_STREAMING Audio Interface Subclass code. Assigned by this specification. See Appendix A.5, "Audio Interface Subclass Codes."
7	bInterfaceProtocol	1	Protocol	IP_VERSION_02_00 Interface Protocol code. Indicates the current version of the specification. See Appendix A.6, "Audio Interface Protocol Codes"
8	iInterface	1	Index	Index of a string descriptor that describes this interface.

4.9.2 Class-Specific AS Interface Descriptor

The **bTerminalLink** field contains the unique Terminal ID of the Input or Output Terminal to which this interface is connected.

The **bmControls** field contains a set of bit pairs, indicating which Controls are present and what their capabilities are. If a Control is present, it must be Host readable. If a certain Control is not present then the bit pair must be set to 0b00. If a Control is present but read-only, the bit pair must be set to 0b01. Since both Active Alternate Setting and Valid Alternate Settings Controls can only be read-only, the value 0b11 is not allowed. The value 0b10 is also not allowed.

The **bFormatType** field indicates which Format Type this interface is using. There are currently four Simple Format Types defined. Type I, II, and III can be used when the interface has a USB isochronous endpoint associated with it. The **bmFormats** field further qualifies which Audio Data Format(s) can be used over the endpoint of this interface. Type IV is reserved for those cases where the interface has no endpoint associated with it. The **bmFormats** field then describes the Audio Data Format that is used on the (external) connection this interface represents.

The different Format Types are further specified in a separate document, *USB Audio Data Formats* that is considered part of this specification.

An Alternate Setting of an interface is allowed to support multiple Audio Data Formats within one Format Type group at the same time. The **bmFormats** bitmap field has a bit set for each Audio Data Format that can be used when communicating with this Alternate Setting of the interface. However, it is only allowed to group related Audio Data Formats (formats that can be handled by one decoder) into the same Alternate Setting and not mix Audio Data Formats that require different decoders.

The **bNrChannels**, **bmChannelConfig** and **iChannelNames** fields together constitute the physical audio channel cluster descriptor. The **bNrChannels** field contains the number of physical channels in the audio data stream. The bmChannelConfig field is a bitmap that indicates which spatial locations are occupied by the physical channels present in the cluster. The iChannelNames field allows for custom-defined extensions.

This specification defines a number of standard Formats, ranging from Mono 8-bit PCM to MPEG2 7.1 encoded audio streams. A complete list of supported Format Types and Audio Data Formats is provided in a separate document, *USB Audio Data Formats*, which is considered part of this specification. Further specific information concerning the Format Type for this interface is reported in a separate type-specific descriptor. See Section 4.9.3, “Class-Specific AS Format Type Descriptor.”

Table 4-27: Class-Specific AS Interface Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	Number	Size of this descriptor in bytes: 16
1	bDescriptorType	1	Constant	CS_INTERFACE descriptor type.
2	bDescriptorSubtype	1	Constant	AS_GENERAL descriptor subtype.
3	bTerminalLink	1	Constant	The Terminal ID of the Terminal to which this interface is connected.
4	bmControls	1	Bitmap	D1..0: Active Alternate Setting Control D3..2: Valid Alternate Settings Control D7..4: Reserved. Must be set to 0.
5	bFormatType	1	Constant	Constant identifying the Format Type the AudioStreaming interface is using.
6	bmFormats	4	Bitmap	The Audio Data Format(s) that can be used to communicate with this interface. See the *USB Audio Data Formats* document for further details.
10	bNrChannels	1	Number	Number of physical channels in the AS Interface audio channel cluster.
11	bmChannelConfig	4	Bitmap	Describes the spatial location of the physical channels.

Offset	Field	Size	Value	Description
15	iChannelNames	1	Index	Index of a string descriptor, describing the name of the first physical channel.

4.9.3 Class-Specific AS Format Type Descriptor

Each Format Type has a specific Format Type descriptor associated with it. This class-specific AS Format Type descriptor follows the class-specific AS interface descriptor and delivers format type-specific information to the Host. The details and layout of this descriptor for each of the supported Format Types is found in the *USB Audio Data Formats* document.

4.9.4 Class-Specific AS Encoder Descriptor

4.9.4.1 Encoder Descriptor

The encoder is uniquely identified by the value in the **bEncoderID** field of the encoder descriptor. No other entity within the AudioStreaming interface may have the same ID. This value must be passed in the Entity ID field (part of the **wIndex** field) of each request that is directed to the encoder.

The **bEncoder** field contains a constant, identifying the encoder type.

The **bmControls** field contains a set of bit pairs, indicating which other Controls are present and what their capabilities are. If a Control is present, it must be Host readable. If a certain Control is not present then the bit pair must be set to 0b00. If a Control is present but read-only, the bit pair must be set to 0b01. If a Control is also Host programmable, the bit pair must be set to 0b11. The value 0b10 is not allowed.

Eight custom Parameter Controls are provided. These can be used to control the encoder in a vendor-specific manner.

The **iParam1** to **iParam8** fields each contain an index to a string descriptor that in turn contains a user-readable description of the purpose or effect of the Control.

Table 4-28: Encoder Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	Number	Size of this descriptor, in bytes: 21
1	bDescriptorType	1	Constant	CS_INTERFACE descriptor type.
2	bDescriptorSubtype	1	Constant	ENCODER descriptor subtype.
3	bEncoderID	1	Constant	Constant uniquely identifying the encoder within the interface. This value is used in all requests to address this encoder.
4	bEncoder	1	Constant	Constant identifying the encoder. See Appendix A.15, “Encoder Type Codes.”

Offset	Field	Size	Value	Description
8	bmControls	4	Bitmap	D1..0: Bit Rate Control D3..2: Quality Control D5..4: VBR Control D7..6: Type Control D9..8: Underflow Control D11..10: Overflow Control D13..12: Encoder Error Control D15..14: Param1 Control D17..16: Param2 Control D19..18: Param3 Control D21..20: Param4 Control D23..22: Param5 Control D25..24: Param6 Control D27..26: Param7 Control D29..28: Param8 Control D31..30: Reserved. Must be set to 0.
12	iParam1	1	Index	Index of a string descriptor, describing the purpose of Param1.
13	iParam2	1	Index	Index of a string descriptor, describing the purpose of Param2.
14	iParam3	1	Index	Index of a string descriptor, describing the purpose of Param3.
15	iParam4	1	Index	Index of a string descriptor, describing the purpose of Param4.
16	iParam5	1	Index	Index of a string descriptor, describing the purpose of Param5.
17	iParam6	1	Index	Index of a string descriptor, describing the purpose of Param6.
18	iParam7	1	Index	Index of a string descriptor, describing the purpose of Param7.
19	iParam8	1	Index	Index of a string descriptor, describing the purpose of Param8.
20	iEncoder	1	Index	Index of a string descriptor, describing the encoder.

4.9.5 Class-Specific AS Decoder Descriptor

4.9.5.1 MPEG Decoder Descriptor

The decoder is uniquely identified by the value in the **bDecoderID** field of the decoder descriptor. No other entity within the AudioStreaming interface may have the same ID. This value must be passed in the Entity ID field (part of the **wIndex** field) of each request that is directed to the decoder.

The **bDecoder** field contains a constant, identifying that this is an MPEG decoder.

The **bmMPEGCapabilities** bitmap field describes the capabilities of the MPEG decoder built into the AudioStreaming interface.

Bits D2..0 of the **bmMPEGCapabilities** field are used to indicate which layers this decoder is capable of processing. The different layers relate to the different algorithms that are used during encoding and decoding.

Bit D3 indicates that the decoder can only process the MPEG-1 base stream. Therefore, only Left and Right channels will be output.

Bit D4 indicates that the decoder can handle MPEG-2 streams that contain two independent stereo pairs instead of the normal 3/2 encoding scheme. This bit is only applicable for MPEG-2 decoders.

Bit D5 indicates that the decoder supports the MPEG dual channel mode. In this case, the MPEG-1 base stream does not contain Left and Right channels of a stereo pair but instead contains two independent mono channels. One of these channels can be selected through the proper request (Dual Channel Control) and reproduced over the Left and Right output channels simultaneously.

Bit D6 indicates that the decoder supports the DVD MPEG-2 augmentation to 7.1 channels instead of the standard 5.1 channels.

Bit D7 indicates that the decoder is capable of processing streams that are encoded using adaptive multi-channel prediction.

Bits D9..8 indicate if the decoder can process embedded multilingual information. Multilingual capabilities can consist of being able to process multilingual information encoded at the same sampling frequency as the main audio channels (D9..8 = 0b01). Some decoders may provide the additional capability to process multilingual information encoded at half the sampling frequency of the main audio channels (D9..8 = 0b11).

Bit 10 indicates the ability of the decoder to handle low sampling frequencies (16 kHz, 22.05 kHz and 24 kHz) besides the standard 32 kHz, 44.1 kHz and 48 kHz sampling frequencies.

Bits D15..11 are reserved for future extensions.

The **bmMPEGFeatures** field indicates compression-related features.

Bits D5..4 report which type of Dynamic Range Control the MPEG decoder supports. Some decoders do not implement DRC (D5..4 = 0b00). If implemented, the DRC can either use the stream embedded gain parameters as is (D5..4 = 0b01) or can provide for additional DRC scaling factors, either a single scaling factor that influences both the boost and cut value simultaneously (D5..4 = 0b10) or a separate scaling factor for the boost and the cut value (D5..4 = 0b11)

All other bits are reserved.

The presence of the Dual Channel Control, Second Stereo Control, Mulitlingual Control, Dynamic Range Control, Scaling Control, and High/Low Scaling Control can be derived from the reported capabilities and features of the MPEG decoder. If a capability or feature is supported, the corresponding Control(s) must also be implemented and must be Host programmable.

The **bmControls** field contains a set of bit pairs, indicating which other Controls are present and what their capabilities are. If a Control is present, it must be Host readable. If a certain Control is not present then the bit pair must be set to 0b00. If a Control is present but read-only, the bit pair must be set to 0b01. If a Control is also Host programmable, the bit pair must be set to 0b11. The value 0b10 is not allowed.

Table 4-29: MPEG Decoder Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	Number	Size of this descriptor, in bytes: 10
1	bDescriptorType	1	Constant	CS_INTERFACE descriptor type.

Offset	Field	Size	Value	Description
2	bDescriptorSubtype	1	Constant	DECODER descriptor subtype.
3	bDecoderID	1	Constant	Constant uniquely identifying the decoder within the interface. This value is used in all requests to address this decoder.
4	bDecoder	1	Constant	MPEG_DECODER. Constant identifying the decoder. See Appendix A.16, “Decoder Type Codes.”
5	bmMPEGCapabilities	2	Bitmap	Bitmap identifying the MPEG capabilities of the decoder. A bit set indicates that the capability is supported: D2..0: Layer support: D0 = Layer I D1 = Layer II D2 = Layer III D3: MPEG-1 only. D4: MPEG-1 dual-channel. D5: MPEG-2 second stereo. D6: MPEG-2 7.1 channel augmentation. D7: Adaptive multi-channel prediction. D9..8: MPEG-2 multilingual support: 00 = Not supported 01 = Supported at Fs 10 = Reserved 11 = Supported at F_s and $½F_s$. D10: Support for $½F_s$. D15..11: Reserved. Must be set to 0.
7	bmMPEGFeatures	1	Bitmap	Bitmap identifying the features the decoder supports. A bit set indicates that the feature is supported: D3..0: Reserved. D5..4: Internal Dynamic Range Control: 00 = not supported. 01 = supported but not scalable. 10 = scalable, common boost and cut scaling value. 11 = scalable, separate boost and cut scaling value. D7..6: Reserved. Must be set to 0.

Offset	Field	Size	Value	Description
8	bmControls	1	Bitmap	D1..0: Underflow Control D3..2: Overflow Control D5..4: Decoder Error Control D7..6: Reserved. Must be set to 0.
9	iDecoder	1	Index	Index of a string descriptor, describing the decoder.

4.9.5.2 AC-3 Decoder Descriptor

The decoder is uniquely identified by the value in the **bDecoderID** field of the decoder descriptor. No other entity within the AudioStreaming interface may have the same ID. This value must be passed in the Entity ID field (part of the **wIndex** field) of each request that is directed to the decoder.

The **bDecoder** field contains a constant, identifying that this is an AC-3 decoder.

The **bmBSID** bitmap field describes which bit stream ID modes this decoder is capable of processing. BSID modes can range from 0 to 31. A bit set indicates that BSID mode [bit_position] is supported. Standard AC-3 decoders must be capable of processing at least BSID modes 0 to 8. Therefore, the lower 9 bits of the **bmBSID** field must be set.

The **bmAC3Features** bitmap field indicates compression-related features.

Bits D3..0 indicate which mode the decoder supports. To ease the design of decoder products, Dolby Digital ICs offer standard operating modes called “Line Mode” and “RF Mode.” These modes are included within the Dolby Digital decoder IC itself, thus greatly simplifying the implementation of dialog normalization, dynamic range control and down-mixing functions, all of which are necessary in Dolby Digital products. Two “Custom Modes” offer additional design flexibility aimed at more esoteric audio products for which additional implementation cost and complexity are not of primary concern.

Bits D5..4. indicate which type of Dynamic Range Control the AC-3 decoder supports. Some decoders do not implement DRC (D5..4 = 0b00). If implemented, the DRC can either use the stream embedded gain parameters as is (D5..4 = 0b01) or can provide for additional DRC scaling factors: either a single scaling factor that influences both the boost and cut value simultaneously (D5..4 = 0b10), or a separate scaling factor for the boost and the cut value (D5..4 = 0b11)

All other bits are reserved.

The presence of the Mode Control, Dynamic Range Control, Scaling Control, and High/Low Scaling Control can be derived from the reported capabilities and features of the AC-3 decoder. If a capability or feature is supported, the corresponding Control(s) must also be implemented and must be Host programmable.

The **bmControls** field contains a set of bit pairs, indicating which other Controls are present and what their capabilities are. If a Control is present, it must be Host readable. If a certain Control is not present then the bit pair must be set to 0b00. If a Control is present but read-only, the bit pair must be set to 0b01. If a Control is also Host programmable, the bit pair must be set to 0b11. The value 0b10 is not allowed.

Table 4-30: AC-3 Decoder Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	Number	Size of this descriptor, in bytes: 12
1	bDescriptorType	1	Constant	CS_INTERFACE descriptor type.

Offset	Field	Size	Value	Description
2	bDescriptorSubtype	1	Constant	DECODER descriptor subtype.
3	bDecoderID	1	Constant	Constant uniquely identifying the decoder within the interface. This value is used in all requests to address this decoder.
4	bDecoder	1	Constant	AC-3_DECODER. Constant identifying the decoder. See Appendix A.16, “Decoder Type Codes.”
5	bmBSID	4	Bitmap	A bit set to 1 indicates that the corresponding BSID mode is supported.
9	bmAC3Features	1	Bitmap	A bit set to 1 indicates that the mentioned feature is supported: D0: RF mode D1: Line mode D2: Custom0 mode D3: Custom1 mode D5..4: Internal Dynamic Range Control: 00 = not supported. 01 = supported but not scalable. 10 = scalable, common boost and cut scaling value. 11 = scalable, separate boost and cut scaling value. D7..6: Reserved. Must be set to 0.
10	bmControls	1	Bitmap	D1..0: Underflow Control D3..2: Overflow Control D5..4: Decoder Error Control D7..6: Reserved. Must be set to 0.
11	iDecoder	1	Index	Index of a string descriptor, describing the decoder.

4.9.5.3 WMA Decoder Descriptor

The decoder is uniquely identified by the value in the **bDecoderID** field of the decoder descriptor. No other entity within the AudioStreaming interface may have the same ID. This value must be passed in the Entity ID field (part of the **wIndex** field) of each request that is directed to the decoder.

The **bDecoder** field contains a constant, identifying that this is a WMA decoder.

The **bmWMAProfile** bitmap field indicates compression-related features. The specific profile supported by the decoder is indicated per the table below. There are four profiles each for WMA and WMA Pro and two profiles for WMA Speech.

The **bmControls** field contains a set of bit pairs, indicating which Controls are present and what their capabilities are. If a Control is present, it must be Host readable. If a certain Control is not present then the bit pair must be set to 0b00. If a Control is present but read-only, the bit pair must be set to 0b01. If a Control is also Host programmable, the bit pair must be set to 0b11. The value 0b10 is not allowed.

Table 4-31: WMA Decoder Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	Number	Size of this descriptor, in bytes: 9
1	bDescriptorType	1	Constant	CS_INTERFACE descriptor type.
2	bDescriptorSubtype	1	Constant	DECODER descriptor subtype.
3	bDecoderID	1	Constant	Constant uniquely identifying the decoder within the interface. This value is used in all requests to address this decoder.
4	bDecoder	1	Constant	WMA_DECODER. Constant identifying the decoder. See Appendix A.16, "Decoder Type Codes."
5	bmWMAProfile	2	Bitmap	A bit set to 1 indicates that the mentioned feature is supported: D0: WMA Profile 1, L1 D1: WMA Profile 2, L2 D2: WMA Profile 3, L3 D3: WMA Profile Other, L D4: WMA Speech 1, S1 D5: WMA Speech 2, S2 D6: WMAPro Profile 1, M1 D7: WMAPro Profile 2, M2 D8: WMAPro Profile 3, M3 D9: WMAPro Profile Other, M D10: WMA lossless decoding is supported D15..11: Reserved. Must be set to 0.
7	bmControls	1	Bitmap	D1..0: Underflow Control D3..2: Overflow Control D5..4: Decoder Error Control D7..6: Reserved. Must be set to 0.
8	iDecoder	1	Index	Index of a string descriptor, describing the decoder.

4.9.5.4 DTS Decoder Descriptor

The decoder is uniquely identified by the value in the **bDecoderID** field of the decoder descriptor. No other entity within the AudioStreaming interface may have the same ID. This value must be passed in the Entity ID field (part of the **wIndex** field) of each request that is directed to the decoder.

The **bDecoder** field contains a constant, identifying that this is a DTS decoder.

The **bmCapabilities** bitmap field describes the capabilities of the DTS decoder. The definitions are as follows:

Bit D0: **Core** – Capable of decoding legacy bit streams of up to 1.5 mbps. This does not fully specify the decoder feature set, but guarantees that the decoder is capable of producing meaningful output from any DTS bit stream that includes Core data.

Bit D1: **Lossless** – Capable of producing meaningful output from a DTS bit stream that contains only lossless encoded data; that is, a DTS bit stream that does not include Core data.

Bit D2: **LBR** – Capable of producing meaningful output from a DTS bit stream that contains only LBR encoded data; that is, a DTS bit stream that does not include Core data.

Bit D3: **MultipleStreamMixing** – Capable of mixing audio from as many as three distinct streams. One or more stream will be the PCM output of a decoder. The third stream must be a PCM stream. The mixing coefficients, if not defaults, must be in the form of dynamic metadata.

Bit D4: **DualDecode** – Capable of simultaneously decoding two separate bit streams. Typically the primary stream will be Core Compatible and the secondary stream will be either Core Compatible or LBR. Multiple Stream Mixing must be enabled in order for Dual Decode to be enabled.

The **bmControls** field contains a set of bit pairs, indicating which Controls are present and what their capabilities are. If a Control is present, it must be Host readable. If a certain Control is not present then the bit pair must be set to 0b00. If a Control is present but read-only, the bit pair must be set to 0b01. If a Control is also Host programmable, the bit pair must be set to 0b11. The value 0b10 is not allowed.

Table 4-32: DTS Decoder Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	Number	Size of this descriptor, in bytes: 8
1	bDescriptorType	1	Constant	CS_INTERFACE descriptor type.
2	bDescriptorSubtype	1	Constant	DECODER descriptor subtype.
3	bDecoderID	1	Constant	Constant uniquely identifying the decoder within the interface. This value is used in all requests to address this decoder.
4	bDecoder	1	Constant	DTS_DECODER. Constant identifying the decoder. See Appendix A.16, “Decoder Type Codes.”
5	bmCapabilities	1	Bitmap	A bit set to 1 indicates that the mentioned feature is supported: D0: Core D1: Lossless D2: LBR D3: MultipleStreamMixing D4: DualDecode D7..5: Reserved. Must be set to 0.
6	bmControls	1	Bitmap	D1..0: Reserved. Must be set to 0b00. D3..2: Underflow Control D5..4: Overflow Control D7..6: Decoder Error Control

Offset	Field	Size	Value	Description
7	iDecoder	1	Index	Index of a string descriptor, describing the decoder.

4.10 AudioStreaming Endpoint Descriptors

The following sections describe all possible endpoint-related descriptors for the AudioStreaming interface.

4.10.1 AS 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.

4.10.1.1 Standard AS Isochronous Audio Data Endpoint Descriptor

The standard AS isochronous audio data endpoint descriptor is identical to the standard endpoint descriptor defined in Section 9.6.6, "Endpoint," of the *USB Specification* and further expanded as defined in the *Universal Serial Bus Class Specification*. 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 must be set to Asynchronous, Adaptive or Synchronous. For further details, refer to Section 5.12.4.1, "Synchronous Type," of the *USB Specification*.

Table 4-33: Standard AS 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, reset to zero D7: Direction. 0 = OUT endpoint 1 = IN endpoint

Offset	Field	Size	Value	Description
3	bmAttributes	1	Bit Map	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	Maximum packet size this endpoint is capable of sending or receiving when this configuration is selected. This is determined by the audio bandwidth constraints of the endpoint.
6	bInterval	1	Number	Interval for polling endpoint for data transfers.

4.10.1.2 Class-Specific AS Isochronous Audio Data Endpoint Descriptor

The **bmAttributes** field contains bit D7 to indicate whether the endpoint always needs USB packets of **wMaxPacketSize** length (D7 = 0b1) or that it can handle short packets (D7 = 0b0). In any case, the endpoint is required to support null packets. This bit must be used by the Host software to determine if the driver should pad all potential short packets (except null packets) with zero bytes to **wMaxPacketSize** length before sending them to an OUT endpoint. Likewise, when receiving data from an IN endpoint, the Host software must be prepared to receive wMaxPacketSize bytes and discard any superfluous zero bytes in the packet.

Note: This bit can only be used for Type II formatted data. The data stream must contain enough information (in a header) to separate the actual data from the padded zero bytes.

The **bmControls** field contains a set of bit pairs, indicating which Controls are present and what their capabilities are. If a Control is present, it must be Host readable. If a certain Control is not present then the bit pair must be set to 0b00. If a Control is present but read-only, the bit pair must be set to 0b01. If a Control is also Host programmable, the bit pair must be set to 0b11. The value 0b10 is not allowed.

The **bLockDelayUnits** and **wLockDelay** fields are used to indicate to the Host how long it takes for the clock recovery circuitry of this endpoint to lock and reliably produce or consume the audio data stream. This information can be used by the Host to take appropriate action so that no meaningful data gets lost during the locking period. (For instance, sending digital silence during lock period)

Depending on the implementation, the locking period can be a fixed amount of time or can be proportional to the sampling frequency. In this case, it usually takes a fixed amount of samples to become locked. To accommodate both cases, the **bLockDelayUnits** field indicates whether the **wLockDelay** field is expressed in time (milliseconds) or number of samples.

Note: Some implementations may use locking strategies that do not have either a fixed time or a fixed number of samples before locking. In this case, a worst case value can be reported back to the Host.

The **bLockDelayUnits** and **wLockDelay** fields are only applicable for synchronous and adaptive endpoints. For asynchronous endpoints, the clock is generated internally in the audio function and is completely independent. In this case, **bLockDelayUnits** and **wLockDelay** must be set to zero.

Table 4-34: Class-Specific AS Isochronous Audio Data Endpoint Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	Number	Size of this descriptor, in bytes: 8
1	bDescriptorType	1	Constant	CS_ENDPOINT descriptor type.
2	bDescriptorSubtype	1	Constant	EP_GENERAL descriptor subtype.
3	bmAttributes	1	Bit Map	Bit D7 indicates a requirement for **wMaxPacketSize** packets. D7: MaxPacketsOnly
4	bmControls	1	Bitmap	D1..0: Pitch Control D3..2: Data Overrun Control D5..4: Data Underrun Control D7..6: Reserved. Must be set to 0.
5	bLockDelayUnits	1	Number	Indicates the units used for the wLockDelay field: 0: Undefined 1: Milliseconds 2: Decoded PCM samples 3..255: Reserved
6	wLockDelay	2	Number	Indicates the time it takes this endpoint to reliably lock its internal clock recovery circuitry. Units used depend on the value of the bLockDelayUnits field.

4.10.2 AS 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.

4.10.2.1 Standard AS Isochronous Feedback Endpoint Descriptor

The isochronous feedback endpoint descriptor is identical to the standard endpoint descriptor defined in Section 9.6.6, “Endpoint,” of the *USB Specification*. The **bmAttributes** field bits are set to reflect the isochronous type and synchronization type of the endpoint.

Table 4-35: Standard AS 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, reset to zero D7: Direction. 0 = OUT endpoint 1 = IN endpoint
3	bmAttributes	1	Bit Map	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.

4.10.2.2 Class-Specific AS Isochronous Feedback Endpoint Descriptor

There is no class-specific AS isochronous feedback endpoint descriptor.

5 Requests

5.1 Standard Requests

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

5.2 Class-Specific Requests

Class-specific requests are used to set and get audio related Controls. These Controls fall into two main groups: those that manipulate the audio function’s Controls, such as volume, tone, selector position, etc. and those that influence data transfer over an isochronous endpoint, such as the current sampling frequency.

- **AudioControl Requests**. Control of an audio function is performed through the manipulation of the attributes of individual Controls that are embedded in the Entities of the audio function. The class-specific AudioControl interface descriptor contains a collection of Entity descriptors, each indicating which Controls are present in the Entity. AudioControl requests are always directed to the single AudioControl interface of the audio function. The request contains enough information (Entity ID, Control Selector, and Channel Number) for the audio function to decide to where a specific request must be routed. The same request layout can be used for vendor-specific requests to Extension Units. However, they are not covered by this specification.
- **AudioStreaming Requests**. Control of the class-specific behavior of an AudioStreaming interface is performed through manipulation of either interface Controls or endpoint Controls. These can be either class-specific (as defined in this specification) or vendor-specific. In either case, the same request layout can be used. AudioStreaming requests are directed to the recipient where the Control resides. This can be either the interface or its associated isochronous endpoint.

The Audio Device Class supports one additional class-specific request:

- **Memory Requests**. Every addressable Entity in the audio function (Clock Entity, Terminal, Unit, Interface, and Endpoint) can expose a memory-mapped interface that provides the means to generically manipulate the Entity. Vendor-specific Control implementations could be based on this type of request.

In principle, all requests are optional. If an audio function does not support a certain request, it must indicate this by stalling the control pipe when that request is issued to the function. However, if a certain Set request is supported, the associated Get request must also be supported. Get requests may be supported without the associated Set request being supported. If interrupts are supported, then all necessary Get requests must be implemented that are required to retrieve the appropriate information from the audio function in response to these interrupts.

The remainder of this section describes the class-specific requests and their characteristics used to manipulate the incorporated Controls.

5.2.1 Control Attributes

Each Control within an Entity can have one or more attributes associated with it. Currently defined attributes for a Control are its:

- Current setting attribute (CUR)
- Range attribute (RANGE)

The CUR attribute is used to manipulate the current actual setting of a Control. The RANGE attribute provides information about the limitations the Control imposes on the allowed settings of the CUR

attribute. The RANGE attribute actually consists of an array of subattributes. The subattributes are Minimum (MIN), Maximum (MAX), and Resolution (RES). They are always manipulated in triplets of the form [MIN, MAX, RES] and cannot be accessed or modified individually. The RANGE attribute supports an array of these triplets so that discontinuous multiple subranges of a Control can be accurately reported. The first element in the array contains the number of subranges the Control supports. Subsequent triplet elements in the array correspond to each of the subranges. The subranges must be ordered in ascending order (from lower values to higher values). Individual subranges cannot overlap (i.e. the MAX value of the previous subrange cannot be equal to the MIN value of the next subrange). If a subrange consists of only a single value, the corresponding triplet must contain that value for both its MIN and MAX subattribute and the RES subattribute must be set to zero.

As an example, consider a (hypothetical) Volume Control that can take the following values for its CUR attribute:

- -∞ dB
- -70 dB to -40 dB in steps of 3 dB
- -40 dB to -20 dB in steps of 2 dB
- -20 dB to 0 dB in steps of 1 dB

One possible layout of the RANGE attribute is then:

RANGE(0) = 3
RANGE(1) = [-70, -40, 3]
RANGE(2) = [-38, -20, 2]
RANGE(3) = [-19, 0, 1]

Another way of representing the same Control is as follows:

RANGE(0) = 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. While it is possible to represent a range as the discrete values that make up that range, this representation is highly discouraged.

5.2.2 Control Request Layout

The Audio Device Class-defined request layout closely follows the standard request layout as defined in the *USB Specification*. The request is used to set or get an attribute of a Control inside an Entity of the audio function. The following table details the request layout.

Table 5-1: Request Layout

bmRequest Type	bRequest	wValue	wIndex	wLength	Data
00100001B 10100001B	CUR RANGE	CS and CN or MCN	Entity ID and Interface	Length of parameter block	Parameter block
00100010B 10100010B			Endpoint		

Bit D7 of the **bmRequestType** field specifies whether this is a Set request (D7 = 0b0) or a Get request (D7 = 0b1). It is a class-specific request (D6..5 = 0b01), directed to either an interface (AudioControl or AudioStreaming) of the audio function (D4..0 = 0b00001) or the isochronous endpoint of an AudioStreaming interface (D4..0 = 0b00010).

The **bRequest** field contains a constant, identifying which attribute of the addressed Control is to be manipulated. Possible attributes for a Control are its:

- Current setting attribute (CUR)
- Range attribute (RANGE)

If the addressed Control does not support modification of a certain attribute, the control pipe must indicate a stall when an attempt is made to modify that attribute. In most cases, only the CUR attribute will be supported for the Set request. However, this specification does not prevent a designer from making the RANGE attribute programmable. For the list of Request constants, refer to Appendix A.14, "Audio Class-Specific Request Codes."

As a general rule, when an attribute value is set, a Control will automatically adjust the passed value to the closest available valid value. This value can be retrieved through a subsequent Get Control request.

The single exception to this rule above is the Copy Protect Control. The control pipe must indicate a stall when the Copy Protect Control cannot honor the Set request exactly.

The **wValue** field specifies the Control Selector (CS) in the high byte and the Channel Number (CN) in the low byte. The Control Selector indicates which type of Control this request is manipulating. The Channel Number (CN) indicates which logical channel of the cluster is to be influenced. If a Control is channel independent, then the Control is considered to be a master Control and the virtual channel zero is used to address it (CN = 0). If the request specifies an unknown or unsupported CS or CN to that Unit, the control pipe must indicate a stall.

There is an exception to the above. If the Mixer Unit Control request wants to address a Mixer Control, it specifies CS = MU_MIXER_CONTROL as the Control Selector in the high byte and the Mixer Control Number (MCN) in the low byte.

When the request addresses an Entity in an interface (**bmRequestType** = 0b00100001 or 10100001), the **wIndex** field specifies the interface in the low byte and the Entity ID (Clock Entity ID, Unit ID, Terminal ID, Encoder ID, or Decoder ID). For addressing the interface itself, an Entity ID of zero must be specified in the high byte.

When the request addresses an endpoint (**bmRequestType** = 0b00100010 or 10100010), the **wIndex** field specifies the endpoint to be addressed in the low byte and zero in the high byte.

The values in **wIndex** must be appropriate to the recipient. Only existing Entities in the audio function or in the AudioStreaming interfaces can be addressed and only appropriate interface or endpoint numbers may be used. If the request specifies an unknown or non-Entity ID or an unknown interface or endpoint number, the control pipe must indicate a stall.

The actual parameter(s) for the Set request are passed in the data stage of the control transfer. The length of the parameter block is indicated in the **wLength** field of the request. The layout of the parameter block is qualified by both the **bRequest** and **wIndex** fields. Refer to the following sections for a detailed description of the parameter block layout for all possible Entities.

The actual parameter(s) for the Get request are returned in the data stage of the control transfer. The length of the parameter block to return is indicated in the **wLength** field of the 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 parameter block is qualified by both the **bRequest** and **wIndex** fields. Refer to the following sections for a detailed description of the parameter block layout for all possible Entities.

5.2.3 Control Request Parameter Block Layout

With a few exceptions, almost all Control requests manipulate a single Control parameter during a Set or Get request. For those requests, the possible parameter block layouts can be divided into three categories, depending on the byte size of the Control's CUR attribute. A CUR attribute's size can be either a single

byte, a word (2 bytes) or a double word (4 bytes). The following paragraphs specify the layout of the CUR and RANGE parameter blocks for the three categories.

For those requests that use a deviating parameter block layout, the actual layout is explicitly defined in the relevant sections.

5.2.3.1 Layout 1 Parameter Block

The parameter block for a 1-byte sized CUR attribute of a Control is as follows:

Table 5-2: 1-byte Control CUR Parameter Block

wLength		1		
Offset	**Field**	**Size**	**Value**	**Description**
0	bCUR	1	Number	The setting for the CUR attribute of the addressed Control

The associated parameter block for the RANGE attribute of that Control is as follows:

Table 5-3: 1-byte Control RANGE Parameter Block

wLength		2+3*n		
Offset	**Field**	**Size**	**Value**	**Description**
0	wNumSubRanges	2	Number	The number of subranges of the addressed Control: n
2	bMIN(1)	1	Number	The setting for the MIN attribute of the first subrange of the addressed Control
3	bMAX(1)	1	Number	The setting for the MAX attribute of the first subrange of the addressed Control
4	bRES(1)	1	Number	The setting for the RES attribute of the first subrange of the addressed Control
…	…	…	…	…
2+3*(n-1)	bMIN(n)	1	Number	The setting for the MIN attribute of the last subrange of the addressed Control
3+3*(n-1)	bMAX(n)	1	Number	The setting for the MAX attribute of the last subrange of the addressed Control
4+3*(n-1)	bRES(n)	1	Number	The setting for the RES attribute of the last subrange of the addressed Control

5.2.3.2 Layout 2 Parameter Block

The parameter block for a 2-byte sized CUR attribute of a Control is as follows:

Table 5-4: 2-byte Control CUR Parameter Block

wLength		2		
Offset	**Field**	**Size**	**Value**	**Description**
0	wCUR	2	Number	The setting for the CUR attribute of the addressed Control

The associated parameter block for the RANGE attribute of that Control is as follows:

Table 5-5: 2-byte Control RANGE Parameter Block

wLength		2+6*n		
Offset	**Field**	**Size**	**Value**	**Description**
0	wNumSubRanges	2	Number	The number of subranges of the addressed Control: n
2	wMIN(1)	2	Number	The setting for the MIN attribute of the first subrange of the addressed Control
4	wMAX(1)	2	Number	The setting for the MAX attribute of the first subrange of the addressed Control
6	wRES(1)	2	Number	The setting for the RES attribute of the first subrange of the addressed Control
…	…	…	…	…
2+6*(n-1)	wMIN(n)	2	Number	The setting for the MIN attribute of the last subrange of the addressed Control
4+6*(n-1)	wMAX(n)	2	Number	The setting for the MAX attribute of the last subrange of the addressed Control
6+6*(n-1)	wRES(n)	2	Number	The setting for the RES attribute of the last subrange of the addressed Control

5.2.3.3 Layout 3 Parameter Block

The parameter block for a 4-byte sized CUR attribute of a Control is as follows:

Table 5-6: 4-byte Control CUR Parameter Block

wLength		4		
Offset	**Field**	**Size**	**Value**	**Description**
0	dCUR	4	Number	The setting for the CUR attribute of the addressed Control

The associated parameter block for the RANGE attribute of that Control is as follows:

Table 5-7: 4-byte Control RANGE Parameter Block

wLength		2+12*n		
Offset	**Field**	**Size**	**Value**	**Description**
0	wNumSubRanges	2	Number	The number of subranges of the addressed Control: n
2	dMIN (1)	4	Number	The setting for the MIN attribute of the first subrange of addressed Control
6	dMAX (1)	4	Number	The setting for the MAX attribute of the first subrange of the addressed Control
10	dRES (1)	4	Number	The setting for the RES attribute of the first subrange of the addressed Control
…	…	…	…	…
2+12*(n-1)	dMIN(n)	4	Number	The setting for the MIN attribute of the last subrange of the addressed Control
6+12*(n-1)	dMAX(n)	4	Number	The setting for the MAX attribute of the last subrange of the addressed Control
10+12*(n-1)	dRES(n)	4	Number	The setting for the RES attribute of the last subrange of addressed Control

5.2.4 Common Controls

The following sections describe a number of Controls that can appear in several Entity types. They are described here only once and a reference to these Control descriptions is provided for all those Entities that can incorporate any of these Controls.

5.2.4.1 Enable Control

The Enable Control is used to either enable the functionality of an Entity or bypass the Entity entirely. In the latter case, default behavior is assumed. The Enable Control must have only the CUR attribute. The value of an Enable Control CUR attribute can be either TRUE or FALSE.

The Control Selector field must be set to XX_ENABLE_CONTROL (where XX must be replaced by the appropriate two-letter abbreviation for the particular Entity) and the Channel Number field must be set to zero (master Control).

The parameter block for this Control request uses Layout 1 (See Section 5.2.3.1, “Layout 1 Parameter Block.”)

5.2.4.2 Mode Select Control

The Mode Select Control is used to change the behavior of the Entity. A Mode Select Control must have only the CUR attribute.The valid range for the CUR attribute is from one to the number of modes, supported by the Entity (reported through the **bNrModes** field of the Entity descriptor).

The Control Selector field must be set to XX_MODE_SELECT_CONTROL (where XX must be replaced by the appropriate two-letter abbreviation for the particular Entity) and the Channel Number field must be set to zero (master Control).

The parameter block for this Control request uses Layout 1 (See Section 5.2.3.1, “Layout 1 Parameter Block.”)

5.2.4.3 Cluster Control

The Cluster Control is used to retrieve the current logical Audio Channel Cluster descriptor from an Entity. This Control only supports the Get request (read-only). A Cluster Control must have only the CUR attribute. The CUR attribute returns a descriptor that is formatted as defined in Section 4.1, “Audio Channel Cluster Descriptor”.

The Control Selector field must be set to XX_CLUSTER_CONTROL (where XX must be replaced by the appropriate two-letter abbreviation for the particular Entity) and the Channel Number field must be set to zero (master Control).

The parameter block for the CUR attribute of the Cluster Control is as follows:

Table 5-8: Cluster Control CUR Parameter Block

wLength		6		
Offset	**Field**	**Size**	**Value**	**Description**
0	bNrChannels	1	Number	Number of logical output channels in the Entity’s output audio channel cluster.
1	bmChannelConfig	4	Bitmap	Describes the spatial location of the logical channels.
5	iChannelNames	1	Index	Index of a string descriptor, describing the name of the first logical channel.

5.2.4.4 Underflow Control

The Underflow Control is used to indicate the occurence of a calculation underflow condition within an Entity *since the last Get Underflow request*. Calculation underflow occurs when an attempt is made to assign a negative value to an unsigned variable. This Control only supports the Get request (read-only). Responding to the Get request returns the CUR attribute, and then clears its value. An Underflow Control must have only the CUR attribute. The value of an Underflow Control CUR attribute must be either TRUE (underflow condition occured) or FALSE (normal).

The Control Selector field must be set to XX_UNDERFLOW_CONTROL (where XX must be replaced by the appropriate two-letter abbreviation for the particular Entity) and the Channel Number field indicates the desired Channel.

The parameter block for this Control request uses Layout 1 (See Section 5.2.3.1, “Layout 1 Parameter Block.”)

5.2.4.5 Overflow Control

The Overflow Control is used to indicate the occurence of a calculation overflow condition within an Entity *since the last Get Overflow request*. Calculation overflow occurs when a value is too positive or too negative to be represented after a signed calculation and when it is too positive after an unsigned calculation. This Control only supports the Get request (read-only). Responding to the Get request returns the CUR attribute, and then clears its value. An Overflow Control must have only the CUR attribute. The value of an Overflow Control CUR attribute must be either TRUE (overflow condition occured) or FALSE (normal).

The Control Selector field must be set to XX_OVERFLOW_CONTROL (where XX must be replaced by the appropriate two-letter abbreviation for the particular Entity) and the Channel Number field indicates the desired Channel.

The parameter block for this Control request uses Layout 1 (See Section 5.2.3.1, “Layout 1 Parameter Block.”)

5.2.4.6 Encoder Error Control

The Encoder Error Control is used to indicate the existence of an error condition within an encoder *since the last Get Encoder Error request*. This Control only supports the Get request (read-only). Responding to the Get request returns the CUR attribute, and then clears its value. An Encoder Error Control must have only the CUR attribute. The settings for the CUR attribute can range from -32768 (0x8000) to 32767 (0x7FFF) in steps of 1 (0x0001).The value 0 represents the case where no error is present. All other values indicate that an error condition exists.

This specification defines a number of possible error codes. Table 5-9, “Error Codes”, enumerates the error codes along with their description.

Table 5-9: Error Codes

Error Code	Description
-32768 .. -1	Reserved for Vendor Extensions.
0	No Error.
1	Out of Memory.
2	Out of Bandwidth.
3	Out of Processing Cycles.
4	General Format Frame Error.
5	Format Frame Too Small.
6	Format Frame Too Large.
7	Bad Data Format.
8	Incorrect Number of Channels.
9	Incorrect Sampling Rate.
10	Unable to Meet Target Bitrate.
11	Inconsistent Set of Parameters.
12	Not Ready.
13	Busy.
14 .. 32767	Reserved for future USB Audio Device Class Extensions.

The Control Selector field must be set to XX_ENCODER_ERROR_CONTROL (where XX must be replaced by the appropriate two-letter abbreviation for the particular Entity) and the Channel Number field must be set to zero (master Control).

The parameter block for this Control request uses Layout 2 (See Section 5.2.3.2, “Layout 2 Parameter Block.”)

5.2.4.7 Decoder Error Control

The Decoder Error Control is used to indicate the existence of an error condition within a decoder *since the last Get Decoder Error request*. This Control only supports the Get request (read-only). Responding to the Get request returns the CUR attribute, and then clears its value. A Decoder Error Control must have only the CUR attribute. The settings for the CUR attribute can range from -32768 (0x8000) to 32767 (0x7FFF) in steps of 1 (0x0001).The value 0 represents the case where no error is present. All other values indicate that an error condition exists.

This specification defines a number of possible error codes. Table 5-9, “Error Codes”, enumerates the error codes along with their description.

The Control Selector field must be set to XX_DECODER_ERROR_CONTROL (where XX must be replaced by the appropriate two-letter abbreviation for the particular Entity) and the Channel Number field must be set to zero (master Control).

The parameter block for this Control request uses Layout 2 (See Section 5.2.3.2, “Layout 2 Parameter Block.”)

5.2.4.8 Latency Control

An audio function must either not support this Control (D1..0 = 0b00 in the bmControls field of the class-specific AudioControl Interface descriptor) or support this read-only Control for *every* Terminal and Unit within the audio function (D1..0 = 0b01 in the bmControls field of the class-specific AudioControl Interface descriptor). Terminal latencies must include all latencies incurred by A/D or D/A converters, encoders, decoders, etc.

The Latency Control is used to accurately report the latency, expressed in nanoseconds, incurred by the addressed Entity. This Control only supports the Get request (read-only). A Latency Control must have only the CUR attribute. The settings for the CUR attribute can range from 0 ns (0x00000000) to 4,294,967,295ns (0xFFFFFFFF) in steps of 1 ns (0x00000001).

The Control Selector field must be set to XX_LATENCY_CONTROL (where XX must be replaced by the appropriate two-letter abbreviation for the particular Entity) and the Channel Number field must be set to zero (master Control).

The parameter block for this Control request uses Layout 3 (See Section 5.2.3.3, “Layout 3 Parameter Block.”)

5.2.5 AudioControl Requests

The following sections describe the possible requests that can be used to manipulate the Audio Controls an audio function exposes through its Entities. The same layout of the parameter blocks is used for both the Set and Get requests.

5.2.5.1 Clock Source Control Request

This request is used to manipulate the Controls inside a Clock Source Entity of the audio function. The exact layout of the request is defined in Section 5.2.2, “Control Request Layout.”

The following paragraphs present a detailed description of all possible Controls a Clock Source Entity can incorporate. For each Control, 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.17.1,”Clock Source Control Selectors.”

5.2.5.1.1 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.

The Sampling Frequency Control must support the CUR and RANGE(MIN, MAX, RES) attributes. The settings for the CUR, MIN, MAX and RES attributes can range from 0 Hz (0x00000000) to 4,294,967,295Hz (0xFFFFFFFF) in steps of 1 Hz (0x00000001).

Note: A discrete list of supported sampling frequencies can be expressed using the method as explained in Section 5.2.1, “Control Attributes”

In many cases, the Clock Source Entity represents a crystal oscillator based generator with a single fixed frequency. In that case, the Set request is not supported.

The Control Selector field must be set to CS_SAM_FREQ_CONTROL and the Channel Number field must be set to zero (master Control).

The parameter block for this Control request uses Layout 3 (See Section 5.2.3.3, “Layout 3 Parameter Block.”)

5.2.5.1.2 Clock Validity Control

The Clock Validity Control is used to indicate if the clock signal that is generated by the Clock Source Entity is valid (stable and reliable). Only the Get request is supported for this Control. The Clock Validity Control must have only the CUR attribute. The value of a Clock Validity Control CUR attribute must be either TRUE (clock valid) or FALSE (clock invalid).

The Control Selector field must be set to CS_CLOCK_VALID_CONTROL and the Channel Number field must be set to zero (master Control).

The parameter block for this Control request uses Layout 1 (See Section 5.2.3.1, “Layout 1 Parameter Block.”)

5.2.5.2 Clock Selector Control Request

This request is used to manipulate the Controls inside a Clock Selector Entity of the audio function. The exact layout of the request is defined in Section 5.2.2, “Control Request Layout.”

The following paragraphs present a detailed description of all possible Controls a Clock Selector Entity can incorporate. For each Control, 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.17.2,”Clock Selector Control Selectors.”

5.2.5.2.1 Clock Selector Control

A Clock Selector Entity represents a multi-input source selector, capable of selecting among a number of clock signals. The valid range for the CUR attribute is from one up to the number of Clock Input Pins of the Clock Selector. This value can be found in the **bNrInPins** field of the Clock Selector descriptor.

The Control Selector field must be set to CX_CLOCK_SELECTOR_CONTROL and the Channel Number field must be set to zero (master Control).

The parameter block for this Control request uses Layout 1 (See Section 5.2.3.1, “Layout 1 Parameter Block.”)

5.2.5.3 Clock Multiplier Control Request

This request is used to manipulate the Controls inside a Clock Multiplier Entity of the audio function. The exact layout of the request is defined in Section 5.2.2, “Control Request Layout.”

The following paragraphs present a detailed description of all possible Controls a Clock Multiplier Entity can incorporate. For each Control, 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.17.3,"Clock Multiplier Control Selectors."

5.2.5.3.1 Numerator Control

A Numerator Control is used to retrieve the factor P by which the incoming sampling clock signal is multiplied. This Control only supports the Get request (read-only). A Numerator Control must have only the CUR attribute. The value of the CUR attribute can range from 1 (0x0001) to 2^{16}-1 (0xFFFF).

The Control Selector field must be set to CM_ NUMERATOR_CONTROL and the Channel Number field must be set to zero (master Control).

The parameter block for this Control request uses Layout 2 (See Section 5.2.3.2, "Layout 2 Parameter Block.")

5.2.5.3.2 Denominator Control

A Denominator Control is used to retrieve the factor Q by which the incoming sampling clock signal is divided. This Control only supports the Get request (read-only). A Denominator Control must have only the CUR attribute. The value of the CUR attribute can range from 1 (0x0001) to 2^{16}-1 (0xFFFF).

The Control Selector field must be set to CM_ DENOMINATOR_CONTROL and the Channel Number field must be set to zero (master Control).

The parameter block for this Control request uses Layout 2 (See Section 5.2.3.2, "Layout 2 Parameter Block.")

5.2.5.4 Terminal Control Request

This request is used to manipulate the Controls inside a Terminal of the audio function. The exact layout of the request is defined in Section 5.2.2, "Control Request Layout."

The following paragraphs present a detailed description of all possible Controls a Terminal can incorporate. For each Control, 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.17.4, "Terminal Control Selectors."

5.2.5.4.1 Copy Protect Control

The Copy Protect Control is used to manipulate the Copy Protection Level (CPL) associated with a particular Terminal. Not all Terminals are required to support this Control. Only the Terminals that represent a connection to the audio function where audio streams enter or leave the audio function in a form that enables lossless duplication should consider Copy Protect Control. Input Terminals in this category should only support the Get Terminal Copy Protect Control request whereas Output Terminals in the same category should only support the Set Terminal Copy Protect Control request.

A Copy Protect Control must have only the CUR attribute. The settings for the CUR attribute range from 0 to 2. This means:

- 0: CPL0: Copying is permitted without restriction. The material is either not copyrighted, or the copyright is not asserted.
- 1: CPL1: One generation of copies may be made. The material is copyright protected and is the original.
- 2: CPL2: The material is copyright protected and no digital copying is permitted.

The control pipe must indicate a stall when the Copy Protect Control cannot honor a Set request exactly.

The Control Selector field must be set to TE_COPY_PROTECT_CONTROL and the Channel Number field must be set to zero (master Control).

The parameter block for this Control request uses Layout 1 (See Section 5.2.3.1, “Layout 1 Parameter Block.”)

5.2.5.4.2 Connector Control

The Connector Control is used to examine the insertion state of connectors that can be associated with the Terminal. This Control only supports the Get request (read-only). A Connector Control must have only the CUR attribute. The CUR attribute returns a descriptor that is formatted as defined in Section 4.1, “Audio Channel Cluster Descriptor”. This descriptor reports all the connectors that are currently inserted. Together with the current cluster descriptor used as a reference, this allows Host software to know what is still connected and – by extension – to determine what is unplugged.

The **bNrChannels** field contains the total number of ‘inserted channels’ (both with predefined and non-predefined spatial locations). A ‘1’ in the **bmChannelConfig** bitmap indicates that the connector corresponding to that physical channel is inserted. A ‘0’ indicates that the connector is currently not inserted. If the **bNrChannels** field contains a number that is larger than the number of bits set in the **bmChannelConfig** field, then this indicates that there are connectors inserted that correspond to non-predefined spatial locations. The array of actual channel names can be retrieved through the **iChannelNames** string index.

The Control Selector field must be set to TE_CONNECTOR_CONTROL and the Channel Number field must be set to zero (master Control).

The parameter block for the CUR attribute of the Connector Control is as follows:

Table 5-10: Connector Control CUR Parameter Block

wLength		6		
Offset	**Field**	**Size**	**Value**	**Description**
0	bNrChannels	1	Number	Total number of physical channels that are currently inserted.
1	bmChannelConfig	4	Bitmap	Describes the predefined spatial locations of the currently inserted channels.
5	iChannelNames	1	Index	Index of a string descriptor, describing the name of the first inserted channel with a non-predefined spatial location.

5.2.5.4.3 Overload Control

The Overload Control is used to indicate the existence of an overload condition within a Terminal. (E.g. overvoltage at the analog line-in connector; thermal overload in powered speakers; etc.) This Control only supports the Get request (read-only). An Overload Control must have only the CUR attribute. The value of an Overload Control CUR attribute must be either TRUE (overload condition exists) or FALSE (normal).

The Control Selector field must be set to TE_OVERLOAD_CONTROL and the Channel Number field indicates the desired Channel.

The parameter block for this Control request uses Layout 1 (See Section 5.2.3.1, “Layout 1 Parameter Block.”)

5.2.5.4.4 Cluster Control

This Control can only be implemented for Input Terminals. See Section 5.2.4.3, “Cluster Control” for a detailed description. XX must be replaced by TE.

5.2.5.4.5 Overflow Control

See Section 5.2.4.5, “Overflow Control” for a detailed description. XX must be replaced by TE.

5.2.5.4.6 Underflow Control

See Section 5.2.4.4, “Underflow Control” for a detailed description. XX must be replaced by TE.

5.2.5.5 Mixer Unit Control Request

This request is used to manipulate the Controls inside a Mixer Unit of the audio function. The exact layout of the request is defined in Section 5.2.2, “Control Request Layout.”

The **wValue** field specifies the Control Selector (CS) in the high byte and the Mixer Control Number (MCN) or the Channel Number (CN) in the low byte. If the CS value indicates that a Mixer Control is addressed (CS = MIXER_CONTROL), then the low byte contains the MCN. The MCN is derived according to the rules established in Section 4.7.2.6, “Mixer Unit Descriptor.”

The following paragraphs present a detailed description of all possible Controls a Mixer Unit can incorporate. For each Control, 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.17.5, “Mixer Control Selectors.”

5.2.5.5.1 Mixer Control

A Mixer Unit consists of a number of Mixer Controls, either programmable or fixed. A Mixer Control must support the CUR and RANGE(MIN, MAX, RES) attributes. The settings for the CUR, MIN, and MAX attributes can range from +127.9961 dB (0x7FFF) down to -127.9961 dB (0x8001) in steps of 1/256 dB or 0.00390625 dB (0x0001). The settings for the RES attribute can only have positive values and range from 1/256 dB (0x0001) to +127.9961 dB (0x7FFF).

In addition, code 0x8000, representing silence (i.e., -∞ dB), must always be implemented. However, it must never be reported as the MIN attribute value.

The Control Selector field must be set to MU_MIXER_CONTROL and the Mixer Control Number field must be set to the MCN of the particular Mixer Control that needs to be addressed.

The parameter block for this Control request uses Layout 2 (See Section 5.2.3.2, “Layout 2 Parameter Block.”)

5.2.5.5.2 Cluster Control

See Section 5.2.4.3, “Cluster Control” for a detailed description. XX must be replaced by MU.

5.2.5.5.3 Underflow Control

See Section5.2.4.4, “Underflow Control” for a detailed description. XX must be replaced by MU.

5.2.5.5.4 Overflow Control

See Section 5.2.4.5, “Overflow Control” for a detailed description. XX must be replaced by MU.

5.2.5.6 Selector Unit Control Request

This request is used to manipulate the Controls inside a Selector Unit of the audio function. The exact layout of the request is defined in Section Section 5.2.2, “Control Request Layout.”

The following paragraphs present a detailed description of all possible Controls a Selector Unit can incorporate. For each Control, 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.17.6, “Selector Control Selectors.”

5.2.5.6.1 Selector Control

A Selector Unit represents a multi-channel source selector, capable of selecting among a number of identically configured audio channel clusters. The valid range for the CUR attribute is from one up to the number of Input Pins of the Selector Unit. This value can be found in the **bNrInPins** field of the Selector Unit descriptor.

The Control Selector field must be set to SU_SELECTOR_CONTROL and the Channel Number field must be set to zero (master Control).

The parameter block for this Control request uses Layout 1 (See Section 5.2.3.1, “Layout 1 Parameter Block.”)

5.2.5.7 Feature Unit Control Request

This request is used to manipulate the Controls inside a Feature Unit of the audio function. The exact layout of the request is defined in Section 5.2.2, “Control Request Layout.”

The following paragraphs present a detailed description of all possible Controls a Feature Unit can incorporate. For each Control, 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.17.7, “Feature Unit Control Selectors.”

5.2.5.7.1 Mute Control

The Mute Control is one of the building blocks of a Feature Unit. A Mute Control must have only the CUR attribute. The value of a Mute Control CUR attribute must be either TRUE (muted) or FALSE (not muted).

A particular Mute Control within a Feature Unit is addressed through the Unit ID and Channel Number fields of the Set/Get Feature Unit Control request. The valid range for the Channel Number field is from zero (the ‘master’ channel) up to the number of logical channels in the audio channel cluster.

The Control Selector field must be set to FU_MUTE_CONTROL and the Channel Number field indicates the desired Channel.

The parameter block for this Control request uses Layout 1 (See Section 5.2.3.1, “Layout 1 Parameter Block.”)

5.2.5.7.2 Volume Control

The Volume Control is one of the building blocks of a Feature Unit. A Volume Control must support the CUR and RANGE(MIN, MAX, RES) attributes. The settings for the CUR, MIN, and MAX attributes can range from +127.9961 dB (0x7FFF) down to -127.9961 dB (0x8001) in steps of 1/256 dB or 0.00390625 dB (0x0001). The settings for the RES attribute can only have positive values and range from 1/256 dB (0x0001) to +127.9961 dB (0x7FFF).

In addition, code 0x8000, representing silence (i.e., -∞ dB), must always be implemented. However, it must never be reported as the MIN attribute value.

A particular Volume Control within a Feature Unit is addressed through the Unit ID and Channel Number fields of the Set/Get Feature Unit Control request. The valid range for the Channel Number field is from zero (the ‘master’ channel) up to the number of logical channels in the audio channel cluster.

The Control Selector field must be set to FU_VOLUME_CONTROL and the Channel Number field indicates the desired Channel.

The parameter block for this Control request uses Layout 2 (See Section 5.2.3.2, “Layout 2 Parameter Block.”)

5.2.5.7.3 Bass Control

The Bass Control is one of the building blocks of a Feature Unit. The Bass Control influences the general Bass behavior of the Feature Unit. A Bass Control must support the CUR and RANGE(MIN, MAX, RES) attributes. The settings for the CUR, MIN, and MAX attributes can range from +31.75 dB (0x7F) down to –32.00 dB (0x80) in steps of 0.25 dB (0x01). The settings for the RES attribute can only have positive values and range from 0.25 dB (0x01) to +31.75 dB (0x7F). Other parameters that also influence the behavior of the Bass Control, such as cut-off frequency, cannot be altered through this request.

A particular Bass Control within a Feature Unit is addressed through the Unit ID and Channel Number fields of the Set/Get Feature Unit Control request. The valid range for the Channel Number field is from zero (the ‘master’ channel) up to the number of logical channels in the audio channel cluster.

The Control Selector field must be set to FU_BASS_CONTROL and the Channel Number field indicates the desired Channel.

The parameter block for this Control request uses Layout 1 (See Section 5.2.3.1, “Layout 1 Parameter Block.”)

5.2.5.7.4 Mid Control

The Mid Control is one of the building blocks of a Feature Unit. The Mid Control influences the general Mid behavior of the Feature Unit. A Mid Control must support the CUR and RANGE(MIN, MAX, RES) attributes. The settings for the CUR, MIN, and MAX attributes can range from +31.75 dB (0x7F) down to –32.00 dB (0x80) in steps of 0.25 dB (0x01). The settings for the RES attribute can only have positive values and range from 0.25 dB (0x01) to +31.75 dB (0x7F). Other parameters that also influence the behavior of the Mid Control, such as cut-off frequency, cannot be altered through this request.

A particular Mid Control within a Feature Unit is addressed through the Unit ID and Channel Number fields of the Set/Get Feature Unit Control request. The valid range for the Channel Number field is from zero (the ‘master’ channel) up to the number of logical channels in the audio channel cluster.

The Control Selector field must be set to FU_MID_CONTROL and the Channel Number field indicates the desired Channel.

The parameter block for this Control request uses Layout 1 (See Section 5.2.3.1, “Layout 1 Parameter Block.”)

5.2.5.7.5 Treble Control

The Treble Control is one of the building blocks of a Feature Unit. The Treble Control influences the general Treble behavior of the Feature Unit. A Treble Control must support the CUR and RANGE(MIN, MAX, RES) attributes. The settings for the CUR, MIN, and MAX attributes can range from +31.75 dB (0x7F) down to –32.00 dB (0x80) in steps of 0.25 dB (0x01). The settings for the RES attribute can only have positive values and range from 0.25 dB (0x01) to +31.75 dB (0x7F). Other parameters that also influence the behavior of the Treble Control, such as cut-off frequency, cannot be altered through this request.

A particular Treble Control within a Feature Unit is addressed through the Unit ID and Channel Number fields of the Set/Get Feature Unit Control request. The valid range for the Channel Number field is from zero (the 'master' channel) up to the number of logical channels in the audio channel cluster.

The Control Selector field must be set to FU_TREBLE_CONTROL and the Channel Number field indicates the desired Channel.

The parameter block for this Control request uses Layout 1 (See Section 5.2.3.1, "Layout 1 Parameter Block.")

5.2.5.7.6 Graphic Equalizer Control

The Graphic Equalizer Control is one of the optional building blocks of a 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 S1.11-1986 standard. Bands are numbered from 14 (center frequency of 25 Hz) up to 43 (center frequency of 20,000 Hz), making a total of 30 possible bands. The following table lists the band numbers and their center frequencies

Table 5-11: Band Numbers and Center Frequencies (ANSI S1.11-1986 Standard)

Band Nr.	Center Freq.	Band Nr.	Center Freq.	Band Nr.	Center Freq.
14	25Hz	24*	250Hz	34	2500Hz
15*	31.5Hz	25	315Hz	35	3150Hz
16	40Hz	26	400Hz	36*	4000Hz
17	50Hz	27*	500Hz	37	5000Hz
18*	63Hz	28	630Hz	38	6300Hz
19	80Hz	29	800Hz	39*	8000Hz
20	100Hz	30*	1000Hz	40	10000Hz
21*	125Hz	31	1250Hz	41	12500Hz
22	160Hz	32	1600Hz	42*	16000Hz
23	200Hz	33*	2000Hz	43	20000Hz

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 Get Control request, 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 Set Control request, 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 must 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 control pipe must indicate a stall.

A Graphic Equalizer Control must support the CUR and RANGE(MIN, MAX, RES) attributes. The settings for the CUR, MIN, and MAX attributes can range from +31.75 dB (0x7F) down to –32.00 dB (0x80) in steps of 0.25 dB (0x01). The settings for the RES attribute can only have positive values and range from 0.25 dB (0x01) to +31.75 dB (0x7F).

A particular Graphic Equalizer Control within a Feature Unit is addressed through the Unit ID and Channel Number fields of the Set/Get Feature Unit Control request. The valid range for the Channel Number field is from zero (the 'master' channel) up to the number of logical channels in the audio channel cluster.

The Control Selector field must be set to FU_GRAPHIC_EQUALIZER_CONTROL and the Channel Number field indicates the desired Channel.

The parameter block for the CUR attribute of the Graphic Equalizer Control is as follows:

Table 5-12: Graphic Equalizer Control CUR Parameter Block

wLength		4+(number of bits set in **bmBandsPresent** : NrBits)		
Offset	**Field**	**Size**	**Value**	**Description**
0	bmBandsPresent	4	Bit Map	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	bCUR(Lowest)	1	Number	The setting for the CUR attribute of the lowest band present: 0x7F: +31.75 dB 0x7E: +31.50 dB … 0x00: 0.00 dB … 0x82: -31.50 dB 0x81: -31.75 dB 0x80: -32.00 dB
…	…	…	…	…
4+(NrBits-1)	bCUR(Highest)	1	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 5-13: Graphic Equalizer Control RANGE Parameter Block

wLength		2+3*n		
Offset	**Field**	**Size**	**Value**	**Description**
0	wNumSubRanges	2	Number	The number of subranges of the Graphic Equalizer Control: n
2	bMIN(1)	1	Number	The setting for the MIN attribute of the first subrange of the Graphic Equalizer Control
3	bMAX(1)	1	Number	The setting for the MAX attribute of the first subrange of the Graphic Equalizer Control

wLength		2+3*n		
Offset	**Field**	**Size**	**Value**	**Description**
4	bRES(1)	1	Number	The setting for the RES attribute of the first subrange of the Graphic Equalizer Control
…	…	…	…	…
2+3*(n-1)	bMIN(n)	1	Number	The setting for the MIN attribute of the last subrange of the Graphic Equalizer Control
3+3*(n-1)	bMAX(n)	1	Number	The setting for the MAX attribute of the last subrange of the Graphic Equalizer Control
4+3*(n-1)	bRES(n)	1	Number	The setting for the RES attribute of the last subrange of the Graphic Equalizer Control

5.2.5.7.7 Automatic Gain Control

The Automatic Gain Control (AGC) is one of the building blocks of a Feature Unit. An Automatic Gain Control must have only the CUR attribute. The value of an Automatic Gain Control CUR attribute must be either TRUE or FALSE.

A particular Automatic Gain Control within a Feature Unit is addressed through the Unit ID and Channel Number fields of the Set/Get Feature Unit Control request. The valid range for the Channel Number field is from zero (the ‘master’ channel) up to the number of logical channels in the audio channel cluster.

The Control Selector field must be set to FU_AUTOMATIC_GAIN _CONTROL and the Channel Number field indicates the desired Channel.

The parameter block for this Control request uses Layout 1 (See Section 5.2.3.1, “Layout 1 Parameter Block.”)

5.2.5.7.8 Delay Control

The Delay Control is one of the building blocks of a Feature Unit. A Delay Control must support the CUR and RANGE(MIN, MAX, RES) attributes. The settings for the CUR, MIN, MAX, and RES attributes are expressed in seconds using a 10.22 format (32 bits). Therefore, they range from zero (0x00000000) to 1023.999999761581s (0xFFFFFFFF) in steps of 1/4194304s (0x00000001).

A particular Delay Control within a Feature Unit is addressed through the Unit ID and Channel Number fields of the Set/Get Feature Unit Control request. The valid range for the Channel Number field is from zero (the ‘master’ channel) up to the number of logical channels in the audio channel cluster.

The Control Selector field must be set to FU_DELAY_CONTROL and the Channel Number field indicates the desired Channel.

The parameter block for this Control request uses Layout 3 (See Section 5.2.3.3, “Layout 3 Parameter Block.”)

5.2.5.7.9 Bass Boost Control

The Bass Boost Control is one of the building blocks of a Feature Unit. A Bass Boost Control must have only the CUR attribute. The position of a Bass Boost Control CUR attribute must be either TRUE or FALSE.

A particular Bass Boost Control within a Feature Unit is addressed through the Unit ID and Channel Number fields of the Set/Get Feature Unit Control request. The valid range for the Channel Number field is from zero (the ‘master’ channel) up to the number of logical channels in the audio channel cluster.

The Control Selector field must be set to FU_BASS_BOOST_CONTROL and the Channel Number field indicates the desired Channel.

The parameter block for this Control request uses Layout 1 (See Section 5.2.3.1, “Layout 1 Parameter Block.”)

5.2.5.7.10 Loudness Control

The Loudness Control is one of the building blocks of a Feature Unit. A Loudness Control must have only the CUR attribute. The value of a Loudness Control CUR attribute must be either TRUE or FALSE.

A particular Loudness Control within a Feature Unit is addressed through the Unit ID and Channel Number fields of the Set/Get Feature Unit Control request. The valid range for the Channel Number field is from zero (the ‘master’ channel) up to the number of logical channels in the audio channel cluster.

The Control Selector field must be set to FU_LOUDNESS_CONTROL and the Channel Number field indicates the desired Channel.

The parameter block for this Control request uses Layout 1 (See Section 5.2.3.1, “Layout 1 Parameter Block.”)

5.2.5.7.11 Input Gain Control

The Input Gain Control is one of the building blocks of a Feature Unit. An Input Gain Control must support the CUR and RANGE(MIN, MAX, RES) attributes. The settings for the CUR, MIN, and MAX attributes can range from +127.9961 dB (0x7FFF) down to -127.9961 dB (0x8001) in steps of 1/256 dB or 0.00390625 dB (0x0001). The settings for the RES attribute can only have positive values and range from 1/256 dB (0x0001) to +127.9961 dB (0x7FFF).

A particular Input Gain Control within a Feature Unit is addressed through the Unit ID and Channel Number fields of the Set/Get Feature Unit Control request. The valid range for the Channel Number field is from zero (the ‘master’ channel) up to the number of logical channels in the audio channel cluster.

The Control Selector field must be set to FU_INPUT_GAIN_CONTROL and the Channel Number field indicates the desired Channel.

The parameter block for this Control request uses Layout 2 (See Section 5.2.3.2, “Layout 2 Parameter Block.”)

5.2.5.7.12 Input Gain Pad Control

The Input Gain Pad Control is one of the building blocks of a Feature Unit. An Input Gain Pad Control must support the CUR and RANGE(MIN, MAX, RES) attributes. The settings for the CUR, MIN, and MAX attributes can range from +127.9961 dB (0x7FFF) down to -127.9961 dB (0x8001) in steps of 1/256 dB or 0.00390625 dB (0x0001). The settings for the RES attribute can only have positive values and range from 1/256 dB (0x0001) to +127.9961 dB (0x7FFF).

A particular Input Gain Pad Control within a Feature Unit is addressed through the Unit ID and Channel Number fields of the Set/Get Feature Unit Control request. The valid range for the Channel Number field is from zero (the ‘master’ channel) up to the number of logical channels in the audio channel cluster.

The Control Selector field must be set to FU_INPUT_GAIN_PAD_CONTROL and the Channel Number field indicates the desired Channel.

The parameter block for this Control request uses Layout 2 (See Section 5.2.3.2, “Layout 2 Parameter Block.”)

5.2.5.7.13 Phase Inverter Control

The Phase Inverter Control is one of the building blocks of a Feature Unit. A Phase Inverter Control must have only the CUR attribute. The value of a Phase Inverter Control CUR attribute must be either TRUE or FALSE.

A particular Phase Inverter Control within a Feature Unit is addressed through the Unit ID and Channel Number fields of the Set/Get Feature Unit Control request. The valid range for the Channel Number field is from zero (the ‘master’ channel) up to the number of logical channels in the audio channel cluster.

The Control Selector field must be set to FU_PHASE_INVERTER_CONTROL and the Channel Number field indicates the desired Channel.

The parameter block for this Control request uses Layout 1 (See Section 5.2.3.1, “Layout 1 Parameter Block.”)

5.2.5.7.14 Underflow Control

See Section5.2.4.4, “Underflow Control” for a detailed description. XX must be replaced by FU.

5.2.5.7.15 Overflow Control

See Section 5.2.4.5, “Overflow Control” for a detailed description. XX must be replaced by FU.

5.2.5.8 Effect Unit Control Request

This request is used to manipulate the Controls inside an Effect Unit of the audio function. The exact layout of the request is defined in Section 5.2.2, “Control Request Layout.”

The following paragraphs present a detailed description of all possible Controls an Effect Unit can incorporate. For each Control, 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.17.8, “Effect Unit Control Selectors.”

5.2.5.8.1 Parametric Equalizer Section Effect Unit

5.2.5.8.1.1 Enable Control

See Section 5.2.4.1, “Enable Control” for a detailed description. XX must be replaced by PE.

5.2.5.8.1.2 Center Frequency Control

The Center Frequency Control is used to manipulate the actual center frequency of the PEQS Effect Unit.

The Center Frequency Control must support the CUR and RANGE(MIN, MAX, RES) attributes. The settings for the CUR, MIN, MAX and RES attributes can range from 0 Hz (0x00000000) to 4,294,967,295Hz (0xFFFFFFFF) in steps of 1 Hz (0x00000001).

Note: A discrete list of supported center frequencies can be expressed using the method as explained in Section 5.2.1, “Control Attributes”

The Control Selector field must be set to PE_CENTER_FREQ_CONTROL and the Channel Number field indicates the desired Channel.

The parameter block for this Control request uses Layout 3 (See Section 5.2.3.3, “Layout 3 Parameter Block.”)

5.2.5.8.1.3 QFactor Control

The QFactor Control is used to manipulate the range of frequencies affected around the center frequency of the PEQS Effect Unit.

The QFactor Control must support the CUR and RANGE(MIN, MAX, RES) attributes. The settings for the CUR, MIN, MAX and RES attributes can range from 0 (0x000.00000 – 12.20 format) to 4,095.999999046326 (0xFFF.FFFFF) in steps of 0.000000953674316406 (0x000.00001).

The Control Selector field must be set to PE_QFACTOR_CONTROL and the Channel Number field indicates the desired Channel.

The parameter block for this Control request uses Layout 3 (See Section 5.2.3.3, “Layout 3 Parameter Block.”)

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.

5.2.5.8.1.4 Gain Control

The Gain Control is used to manipulate the amount of gain or attenuation at the center frequency of the PEQS Effect Unit. A Gain Control must support the CUR and RANGE(MIN, MAX, RES) attributes. The settings for the CUR, MIN, and MAX attributes can range from +127.9961 dB (0x7FFF) down to -127.9961 dB (0x8001) in steps of 1/256 dB or 0.00390625 dB (0x0001). The settings for the RES attribute can only have positive values and range from 1/256 dB (0x0001) to +127.9961 dB (0x7FFF).

The Control Selector field must be set to PE_GAIN_CONTROL and the Channel Number field indicates the desired Channel.

The parameter block for this Control request uses Layout 2 (See Section 5.2.3.2, “Layout 2 Parameter Block.”)

5.2.5.8.1.5 Underflow Control

See Section5.2.4.4, “Underflow Control” for a detailed description. XX must be replaced by PE.

5.2.5.8.1.6 Overflow Control

See Section 5.2.4.5, “Overflow Control” for a detailed description. XX must be replaced by PE.

5.2.5.8.2 Reverberation Effect Unit

5.2.5.8.2.1 Enable Control

See Section 5.2.4.1, “Enable Control” for a detailed description. XX must be replaced by RV.

5.2.5.8.2.2 Type Control

The Type Control is a macro parameter that allows global settings of reverb parameters within the Reverberation Effect Unit. When a certain reverb type is selected, each reverb parameter will be set to the most suitable value. The Type Control must support the CUR and RANGE(MIN, MAX, RES) attributes. The valid range for the CUR, MIN, and MAX attributes is from zero to 255. The RES attribute can only have a value of one.

The CUR attribute subrange from 0 to 7 has predefined behavior:

- 0: Room 1 – simulates the reverberation of a small room.
- 1: Room 2 – simulates the reverberation of a medium room.
- 2: Room 3 – simulates the reverberation of a large room.
- 3: Hall 1 – simulates the reverberation of a medium concert hall.
- 4: Hall 2 – simulates the reverberation of a large concert hall.
- 5: Plate – simulates a plate reverberation (a studio device using a metal plate).
- 6: Delay – conventional delay that produces echo effects.
- 7: Panning Delay – special delay in which the delayed sounds move left and right.

The Control Selector field must be set to RV_TYPE_CONTROL and the Channel Number field indicates the desired Channel.

The parameter block for this Control request uses Layout 1 (See Section 5.2.3.1, “Layout 1 Parameter Block.”)

5.2.5.8.2.3 Level Control

The Level Control is used to set the amount of reverberant sound introduced by the Reverberation Effect Unit. The Level Control must support the CUR and RANGE(MIN, MAX, RES) attributes. The valid range for the CUR, MIN, MAX, and RES attributes is from zero to 255%, compared to the level of the original signal.

The Control Selector field must be set to RV_LEVEL_CONTROL and the Channel Number field indicates the desired Channel.

The parameter block for this Control request uses Layout 1 (See Section 5.2.3.1, “Layout 1 Parameter Block.”)

5.2.5.8.2.4 Time Control

The Time Control is used to set the time over which the reverberation, introduced by the Reverberation Effect Unit, will continue. The Time Control must support the CUR and RANGE(MIN, MAX, RES) attributes. The settings for the CUR, MIN, MAX, and RES attributes can range from zero (0x0000) to 255.9961 s (0xFFFF) in steps of 1/256 s or 0.00390625 s (0x0001).

The Control Selector field must be set to RV_TIME_CONTROL and the Channel Number field indicates the desired Channel.

The parameter block for this Control request uses Layout 2 (See Section 5.2.3.2, “Layout 2 Parameter Block”

5.2.5.8.2.5 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 sets the way in which delay repeats. The Delay Feedback Control range must support the CUR and RANGE(MIN, MAX, RES) attributes. The valid range for the CUR, MIN, MAX, and RES attributes is from zero to 255%. Higher values result in more delay repeats.

Note: In practice, the delay feedback amount should be limited to 75% to avoid unexpected feedback distortion and continuous delay loop.

The Control Selector field must be set to RV_FEEDBACK_CONTROL and the Channel Number field indicates the desired Channel.

The parameter block for this Control request uses Layout 1 (See Section 5.2.3.1, “Layout 1 Parameter Block.”)

5.2.5.8.2.6 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 by the Reverberation Effect Unit. The Pre-Delay Control must support the CUR and RANGE(MIN, MAX, RES) attributes. The settings for the CUR, MIN, MAX, and RES attributes can range from zero (0x0000) to 65535ms (0xFFFF) in steps of 1ms (0x0001).

The Control Selector field must be set to RV_PREDELAY_CONTROL and the Channel Number field indicates the desired Channel.

The parameter block for this Control request uses Layout 2 (See Section 5.2.3.2, “Layout 2 Parameter Block.”)

5.2.5.8.2.7 Density Control

The Density Control is used to set the density of reflections in the reverberant sound introduced by the Reverberation Effect Unit. The Density Control must support the CUR and RANGE(MIN, MAX, RES) attributes. The valid range for the CUR, MIN, MAX, and RES attributes is from zero (0x00) to 100 (0x64) in steps of 1 (0x01).

The Control Selector field must be set to RV_DENSITY_CONTROL and the Channel Number field indicates the desired Channel.

The parameter block for this Control request uses Layout 1 (See Section 5.2.3.1, “Layout 1 Parameter Block.”)

5.2.5.8.2.8 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 by the Reverberation Effect Unit. The Hi-Freq Roll-Off Control must support the CUR and RANGE(MIN, MAX, RES) attributes. The settings for the CUR, MIN, MAX and RES attributes can range from 0 Hz (0x00000000) to 4,294,967,295Hz (0xFFFFFFFF) in steps of 1 Hz (0x00000001).

The Control Selector field must be set to RV_HIFREQ_ROLLOFF_CONTROL and the Channel Number field indicates the desired Channel.

The parameter block for this Control request uses Layout 3 (See Section 5.2.3.3, “Layout 3 Parameter Block.”)

5.2.5.8.2.9 Underflow Control

See Section5.2.4.4, “Underflow Control” for a detailed description. XX must be replaced by RV.

5.2.5.8.2.10 Overflow Control

See Section 5.2.4.5, “Overflow Control” for a detailed description. XX must be replaced by RV.

5.2.5.8.3 Modulation Delay Effect Unit

5.2.5.8.3.1 Enable Control

See Section 5.2.4.1, “Enable Control” for a detailed description. XX must be replaced by MD.

5.2.5.8.3.2 Balance Control

The Balance Control is used to set the amount of effect sound introduced by the Modulation Delay Effect Unit. The Balance Control must support the CUR and RANGE(MIN, MAX, RES) attributes. The valid range for the CUR, MIN, MAX attributes is from -100 to +100. A setting of -100 results in 100% of original signal and 0% of effected signal. A setting of zero 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. The settings for the RES attribute can only have positive values and range from 1% (0x01) to 100% (0x64).

The Control Selector field must be set to MD_BALANCE_CONTROL and the Channel Number field indicates the desired Channel.

The parameter block for this Control request uses Layout 1 (See Section 5.2.3.1, “Layout 1 Parameter Block.”)

5.2.5.8.3.3 Rate Control

The Rate Control is used to set the speed (frequency) of the modulator of the delay time introduced by the Modulation Delay Effect Unit. Higher values result in faster modulation. The Rate Control must support the CUR and RANGE(MIN, MAX, RES) attributes. The settings for the CUR, MIN, MAX, and RES attributes can range from zero (0x0000) to 255.9961 Hz (0xFFFF) in steps of 1/256 Hz or 0.00390625 Hz (0x0001).

The Control Selector field must be set to MD_RATE_CONTROL and the Channel Number field indicates the desired Channel.

The parameter block for this Control request uses Layout 2 (See Section 5.2.3.2, “Layout 2 Parameter Block.”)

5.2.5.8.3.4 Depth Control

The Depth Control is used to set the depth at which the effect sound introduced by the Modulation Delay Effect Unit is modulated. Higher values result in deeper modulation. The Depth Control must support the CUR and RANGE(MIN, MAX, RES) attributes. The settings for the CUR, MIN, MAX, and RES attributes can range from zero (0x0000) to 255.9961 ms (0xFFFF) in steps of 1/256 ms or 0.00390625 ms (0x0001).

The Control Selector field must be set to MD_DEPTH_CONTROL and the Channel Number field indicates the desired Channel.

The parameter block for this Control request uses Layout 2 (See Section 5.2.3.2, “Layout 2 Parameter Block.”)

5.2.5.8.3.5 Time Control

The Time Control is used to set the length of the delay time introduced by the Modulation Delay Efect Unit. Higher values result in longer times. The Time Control must support the CUR and RANGE(MIN, MAX, RES) attributes. The settings for the CUR, MIN, MAX, and RES attributes can range from zero (0x0000) to 255.9961 ms (0xFFFF) in steps of 1/256 ms or 0.00390625 ms (0x0001).

The Control Selector field must be set to MD_TIME_CONTROL and the Channel Number field indicates the desired Channel.

The parameter block for this Control request uses Layout 2 (See Section 5.2.3.2, “Layout 2 Parameter Block.”)

5.2.5.8.3.6 Feedback Level Control

The Feedback Level Control is used to set the level at which the effected (delayed) sound introduced by the Modulation Delay Effect Unit is mixed back into to its own input. Higher values result in higher level of feedback. The Feedback Level Control range must support the CUR and RANGE(MIN, MAX, RES) attributes. The valid range for the CUR, MIN, MAX, and RES attributes is from zero to 255%. Higher values result in more delay repeats.

Note: In practice, the modulation delay feedback amount should be limited to 75% to avoid unexpected feedback distortion and continuous delay loop.

The Control Selector field must be set to MD_FEEDBACK_CONTROL and the Channel Number field indicates the desired Channel.

The parameter block for this Control request uses Layout 1 (See Section 5.2.3.1, “Layout 1 Parameter Block.”)

5.2.5.8.3.7 Underflow Control

See Section5.2.4.4, “Underflow Control” for a detailed description. XX must be replaced by MD.

5.2.5.8.3.8 Overflow Control

See Section 5.2.4.5, “Overflow Control” for a detailed description. XX must be replaced by MD.

5.2.5.8.4 Dynamic Range Compressor Effect Unit

5.2.5.8.4.1 Enable Control

See Section 5.2.4.1, “Enable Control” for a detailed description. XX must be replaced by DR.

5.2.5.8.4.2 Compression Ratio Control

The Compression Ratio Control is used to influence the slope of the active part of the static input-to-output transfer characteristic of the Dynamic Range Compressor Processing Unit. The Compression Ratio Control must support the CUR and RANGE(MIN, MAX, RES) attributes. The valid range for the CUR, MIN, MAX, and RES attributes is from zero (0x0000) to 255.9961 (0xFFFF) in steps of 1/256 or 0.00390625 (0x0001).

The Control Selector field must be set to DR_COMPRESSION_RATIO_CONTROL and the Channel Number field indicates the desired Channel.

The parameter block for this Control request uses Layout 2 (See Section 5.2.3.2, “Layout 2 Parameter Block.”)

5.2.5.8.4.3 MaxAmpl Control

The MaxAmpl Control is used to set the upper boundary of the active input range of the compressor. The MaxAmpl Control must support the CUR and RANGE(MIN, MAX, RES) attributes. The settings for the CUR, MIN, and MAX attributes can range from -128.0000 dB (0x8000) to 127.9961 dB (0x7FFF) in steps of 1/256 dB or 0.00390625 dB (0x0001). The settings for the RES attribute can only have positive values and range from 1/256 dB (0x0001) to +127.9961 dB (0x7FFF).

The Control Selector field must be set to DR_MAXAMPL_CONTROL and the Channel Number field indicates the desired Channel.

The parameter block for this Control request uses Layout 2 (See Section 5.2.3.2, “Layout 2 Parameter Block.”)

5.2.5.8.4.4 Threshold Control

The Threshold Control is used to set the lower boundary of the active input range of the compressor. The Threshold Control must support the CUR and RANGE(MIN, MAX, RES) attributes. The settings for the CUR, MIN, and MAX attributes can range from -128.0000 dB (0x8000) to 127.9961 dB (0x7FFF) in steps of 1/256 dB or 0.00390625 dB (0x0001). The settings for the RES attribute can only have positive values and range from 1/256 dB (0x0001) to +127.9961 dB (0x7FFF).

The Control Selector field must be set to DR_TRESHOLD_CONTROL and the Channel Number field indicates the desired Channel.

The parameter block for this Control request uses Layout 2 (See Section 5.2.3.2, “Layout 2 Parameter Block.”)

5.2.5.8.4.5 Attack Time Control

The Attack Time Control is used to determine the response of the compressor to a step increase in the input signal level. The Attack Time Control must support the CUR and RANGE(MIN, MAX, RES) attributes. The settings for the CUR, MIN, MAX, and RES attributes can range from zero (0x0000) to 255.9961 ms (0xFFFF) in steps of 1/256 ms or 0.00390625 ms (0x0001).

The Control Selector field must be set to DR_ATTACK_TIME_CONTROL and the Channel Number field indicates the desired Channel.

The parameter block for this Control request uses Layout 2 (See Section 5.2.3.2, “Layout 2 Parameter Block.”)

5.2.5.8.4.6 Release Time Control

The Release Time Control is used to determine the recovery response of the compressor to a step decrease in the input signal level. The Release Time Control must support the CUR and RANGE(MIN, MAX, RES) attributes. The settings for the CUR, MIN, MAX, and RES attributes can range from zero (0x0000) to 255.9961 ms (0xFFFF) in steps of 1/256 ms or 0.00390625 ms (0x0001).

The Control Selector field must be set to DR_RELEASE_TIME_CONTROL and the Channel Number field indicates the desired Channel.

The parameter block for this Control request uses Layout 2 (See Section 5.2.3.2, “Layout 2 Parameter Block.”)

5.2.5.8.4.7 Underflow Control

See Section5.2.4.4, “Underflow Control” for a detailed description. XX must be replaced by DR.

5.2.5.8.4.8 Overflow Control

See Section 5.2.4.5, “Overflow Control” for a detailed description. XX must be replaced by DR.

5.2.5.9 Processing Unit Control Request

This request is used to manipulate the Controls inside a Processing Unit of the audio function. The exact layout of the request is defined in Section 5.2.2, “Control Request Layout.”

The following paragraphs present a detailed description of all possible Controls a Processing Unit can incorporate. For each Control, 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.17.9, “Processing Unit Control Selectors.”

5.2.5.9.1 Up/Down-mix Processing Unit

5.2.5.9.1.1 Enable Control

See Section 5.2.4.1, “Enable Control” for a detailed description. XX must be replaced by UD.

5.2.5.9.1.2 Mode Select Control

See Section 5.2.4.2, “Mode Select Control” for a detailed description. XX must be replaced by UD.

5.2.5.9.1.3 Cluster Control

See Section 5.2.4.3, “Cluster Control” for a detailed description. XX must be replaced by UD.

5.2.5.9.1.4 Underflow Control

See Section5.2.4.4, “Underflow Control” for a detailed description. XX must be replaced by UD.

5.2.5.9.1.5 Overflow Control

See Section 5.2.4.5, “Overflow Control” for a detailed description. XX must be replaced by UD.

5.2.5.9.2 Dolby Prologic Processing Unit

5.2.5.9.2.1 Enable Control

See Section 5.2.4.1, “Enable Control” for a detailed description. XX must be replaced by DP.

5.2.5.9.2.2 Mode Select Control

See Section 5.2.4.2, “Mode Select Control” for a detailed description. XX must be replaced by DP.

5.2.5.9.2.3 Cluster Control

See Section 5.2.4.3, “Cluster Control” for a detailed description. XX must be replaced by DP.

5.2.5.9.2.4 Underflow Control

See Section5.2.4.4, “Underflow Control” for a detailed description. XX must be replaced by DP.

5.2.5.9.2.5 Overflow Control

See Section 5.2.4.5, “Overflow Control” for a detailed description. XX must be replaced by DP.

5.2.5.9.3 Stereo Extender Processing Unit

5.2.5.9.3.1 Enable Control

See Section 5.2.4.1, “Enable Control” for a detailed description. XX must be replaced by ST.

5.2.5.9.3.2 Width Control

The Width Control is used to change the spatial appearance of the stereo image, produced by the Stereo Extender Processing Unit. The Width Control must support the CUR and RANGE(MIN, MAX, RES) attributes. The valid range for the CUR, MIN, MAX, and RES attributes is from zero to 255.

The Control Selector field must be set to ST_EXT_WIDTH_CONTROL and the Channel Number field must be set to zero (master Control).

The parameter block for this Control request uses Layout 1 (See Section 5.2.3.1, “Layout 1 Parameter Block.”)

5.2.5.9.3.3 Cluster Control

See Section 5.2.4.3, “Cluster Control” for a detailed description. XX must be replaced by ST.

5.2.5.9.3.4 Underflow Control

See Section5.2.4.4, “Underflow Control” for a detailed description. XX must be replaced by ST.

5.2.5.9.3.5 Overflow Control

See Section 5.2.4.5, “Overflow Control” for a detailed description. XX must be replaced by ST.

5.2.5.10 Extension Unit Control Requests

Because this specification has no knowledge about the inner workings of an Extension Unit, it is impossible to define requests that are able to manipulate specific Extension Unit Controls. However, this specification defines a number of Controls an Extension Unit must or can support.

This request is used to manipulate the Controls inside an Extension Unit of the audio function. The exact layout of the request is defined in Section 5.2.2, “Control Request Layout.”

The following paragraphs present a detailed description of all possible Controls a Processing Unit can incorporate. For each Control, 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 an Extension Unit leads to a control pipe stall. The Control Selector codes are defined in Appendix A.17.10, “Extension Unit Control Selectors.”

5.2.5.10.1 Enable Control

See Section 5.2.4.1, “Enable Control” for a detailed description. XX must be set to XU.

5.2.5.10.2 Cluster Control

See Section 5.2.4.3, “Cluster Control” for a detailed description. XX must be set to XU.

5.2.5.10.3 Underflow Control

See Section5.2.4.4, “Underflow Control” for a detailed description. XX must be replaced by XU.

5.2.5.10.4 Overflow Control

See Section 5.2.4.5, “Overflow Control” for a detailed description. XX must be replaced by XU.

5.2.6 AudioStreaming Requests

The following sections describe the requests an audio function can support for its AudioStreaming interfaces. The same layout of the parameter blocks is used for both the Set and Get requests.

AudioStreaming requests can be directed either to the AudioStreaming interface or to the associated isochronous data endpoint, depending on the location of the Control to be manipulated.

5.2.6.1 Interface Control Request

This request is used to manipulate the Controls inside an AudioStreaming interface of the audio function. The exact layout of the request is defined in Section 5.2.2, “Control Request Layout.”

The following paragraphs present a detailed description of all possible Controls an AudioStreaming Interface can incorporate. For each Control, 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.17.11, “AudioStreaming Interface Control Selectors.”

5.2.6.1.1 Active Alternate Setting Control

This Control is used to inform Host software which Alternate Setting of an AudioStreaming interface is currently active. The main purpose of this Control is to notify the Host (through an interrupt) that the last selected Alternate Setting is no longer valid. There can be a variety of reasons why this could happen. For instance, increasing the sampling frequency of a Clock Source Entity might 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. The Host software must then take appropriate action to reactivate the interface by switching to a valid Alternate Setting. This Control only supports the Get request (read-only) and always provides the currently active Alternate Setting for the interface. An Active Alternate Setting Control must have only the CUR attribute. The value of an Active Alternate Setting Control CUR attribute must only be either the last set Alternate Setting or zero.

The Control Selector field must be set to AS_ACT_ALT_SETTING_CONTROL and the Channel Number field must be set to zero (master Control).

The parameter block for this Control request uses Layout 1 (See Section 5.2.3.1, “Layout 1 Parameter Block.”)

5.2.6.1.2 Valid Alternate Settings Control

This Control is used to inform Host software what the currently possible valid Alternate Settings are for an AudioStreaming interface. This Control only supports the Get request (read-only) and always provides a list of currently valid Alternate Settings for the interface. A Valid Alternate Settings Control must have only the CUR attribute. The value of a Valid Alternate Setting Control CUR attribute is a bitmap that contains a bit for each possible active Alternate Setting.

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 must 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 must be set to 0. An attempt to set the interface to an invalid Alternate Setting (through the standard Set Interface request) will result in a control pipe stall.

The Control Selector field must be set to AS_VAL_ALT_SETTINGS_CONTROL and the Channel Number field must be set to zero (master Control).

The parameter block for the CUR attribute of the Valid Alternate Settings Control is as follows:

Table 5-14: Valid Alternate Settings Control CUR Parameter Block

wLength		1+n		
Offset	**Field**	**Size**	**Value**	**Description**
0	bControlSize	1	Number	Indicates the number of bytes in the bmValidAltSettings bit map: n
1	bmValidAltSettings	n	Bit Map	Each set bit represents a currently valid Alternate Setting for the interface.

5.2.6.1.3 Audio Data Format Control

The Audio Data Format Control is used to indicate which Audio Data Format is currently being used by the AduioStreaming interface. Only the Get request is supported for this Control. The Audio Data Format

Control must have only the CUR attribute. The returned value for the Audio Data Format Control CUR attribute follows the definition of the **bmFormats** field in the class-specific AS interface descriptor. Only one bit can be set in the bitmap, indicating exactly which Audio Data Format is being used by the interface.

The Control Selector field must be set to AS_AUDIO_DATA_FORMAT_CONTROL and the Channel Number field must be set to zero (master Control).

The parameter block for this Control request uses Layout 3 (See Section 5.2.3.3, “Layout 3 Parameter Block.”)

5.2.6.2 Encoder Control Request

This Control request can only be implemented in AudioStreaming interfaces that carry audio data streams that are flowing out of the audio function. Depending on the Audio Data Format used by the interface (reflected in the **bFormatType** and **bmFormats** fields of the class-specific AS interface descriptor), the set of requests to control the encoder may vary.

This request is used to manipulate the Encoder Controls inside an AudioStreaming interface of the audio function. The exact layout of the request is defined in Section 5.2.2, “Control Request Layout.”

The following paragraphs present a detailed description of all possible Encoder Controls an AudioStreaming interface can incorporate. For each Control, 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.17.12, “Encoder Control Selectors.”

5.2.6.2.1 Bit Rate Control

The Bit Rate Control is used to manipulate the output bit rate of the encoder.

The Bit Rate Control must support the CUR and RANGE(MIN, MAX, RES) attributes. The settings for the CUR, MIN, MAX and RES attributes can range from 0 bits/s (0x00000000) to 4,294,967,295 bits/s (0xFFFFFFFF) in steps of 1 bit/s (0x00000001).

Note: A discrete list of supported bit rates can be expressed using the method as explained in Section 5.2.1, “Control Attributes”

The Control Selector field must be set to EN_BIT_RATE_CONTROL and the Channel Number field must be set to zero (master Control).

The parameter block for this Control request uses Layout 3 (See Section 5.2.3.3, “Layout 3 Parameter Block.”)

5.2.6.2.2 Quality Control

The Quality Control is used to set the quality level of the encoder. This Control acts independently of the Bit Rate Control. It controls the quality of the generated bit stream. Lower quality typically requires less processing power and produces less latency. Higher quality typically requires more processing power and results in more latency. The Quality Control must support the CUR and RANGE(MIN, MAX, RES) attributes. The valid range for the CUR, MIN, MAX, and RES attributes is from zero – lowest quality (0x00) to 100 – highest quality (0x64) in steps of 1 (0x01).

Note: A discrete list of supported quality levels can be expressed using the method as explained in Section 5.2.1, “Control Attributes”

The Control Selector field must be set to EN_QUALITY_CONTROL and the Channel Number field must be set to zero (master Control).

The parameter block for this Control request uses Layout 1 (See Section 5.2.3.1, “Layout 1 Parameter Block.”)

5.2.6.2.3 VBR Control

The VBR Control is used to select between constant bitrate mode and variable bitrate mode. The VBR Control must have only the CUR attribute. The value of a VBR Control CUR attribute must be either TRUE (variable bitrate mode selected) or FALSE (constant bitrate mode selected).

The Control Selector field must be set to EN_VBR_CONTROL and the Channel Number field must be set to zero (master Control).

The parameter block for this Control request uses Layout 1 (See Section 5.2.3.1, “Layout 1 Parameter Block.”)

5.2.6.2.4 Type Control

The Type Control is used to select between different encoder optimization modes, based on the type of audio that is presented to the encoder. Nine optimization modes are currently defined. The Type Control must have only the CUR attribute. The settings for the CUR attribute range from 1 to 9. This means:

- 1: UNSPECIFIED: The nature, quality and number of channels of the audio is unspecified and can change over time.
- 2: SPEECH: High-quality spoken audio.
- 3: TELEPHONY: Low-quality or limited bandwidth spoken audio.
- 4: MONOPHONIC MUSIC: Monophonic music.
- 5: POLYPHONIC MUSIC: Polyphonic music.
- 6: MOVIE: Multi-channel high dynamic range movie soundtracks.
- 7: GAME: Multi-channel high dynamic range game audio.
- 8: LOSSLESS: Encoder operates in a lossless mode.
- 9: OTHER: Vendor-defined.

The Control Selector field must be set to EN_TYPE_CONTROL and the Channel Number field must be set to zero (master Control).

The parameter block for this Control request uses Layout 1 (See Section 5.2.3.1, “Layout 1 Parameter Block.”)

5.2.6.2.5 Underflow Control

See Section5.2.4.4, “Underflow Control” for a detailed description. XX must be replaced by EN.

5.2.6.2.6 Overflow Control

See Section 5.2.4.5, “Overflow Control” for a detailed description. XX must be replaced by EN.

5.2.6.2.7 Encoder Error Control

See Section 5.2.4.6, “Encoder Error Control” for a detailed description. XX must be replaced by EN.

5.2.6.2.8 Param<X> Control

The Param<X> Control is used to manipulate a vendor-specific parameter of the encoder where <X> represents a value from 1 to 8.

The Param<X> Control must support the CUR and RANGE(MIN, MAX, RES) attributes. The settings for the CUR, MIN, MAX and RES attributes can range from -2,147,483,648 (0x80000000) to 2,147,483,647 (0x7FFFFFFF) in steps of 1 (0x00000001).

Note: A discrete list of supported values can be expressed using the method as explained in Section 5.2.1, “Control Attributes”

The Control Selector field must be set to EN_PARAM<X>_CONTROL and the Channel Number field must be set to zero (master Control).

The parameter block for this Control request uses Layout 3 (See Section 5.2.3.3, “Layout 3 Parameter Block.”)

5.2.6.3 Decoder Control Request

This Control request can only be implemented in AudioStreaming interfaces that carry audio data streams that are flowing into the audio function. Depending on the Audio Data Format used by the interface (reflected in the **bFormatType** and **bmFormats** fields of the class-specific AS interface descriptor), the set of requests to control the decoder may vary. As an example, consider an interface capable of accepting AC-3 encoded data streams. Such an interface obviously incorporates an AC-3 decoder to convert the incoming data stream into the logical channels that eventually enter the audio function through the associated Input Terminal. The same is true for an interface capable of accepting MPEG-2 encoded data streams. However, the parameter set needed to control the AC-3 decoder differs substantially from the parameter set needed to control the MPEG-2 decoder.

5.2.6.3.1 MPEG Decoder Control Request

This request is used to manipulate the MPEG Decoder Controls inside an AudioStreaming interface of the audio function. The exact layout of the request is defined in Section 5.2.2, “Control Request Layout.”

The following paragraphs present a detailed description of all possible MPEG Decoder Controls an AudioStreaming interface can incorporate. For each Control, 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.17.13.1, “MPEG Decoder Control Selectors.”

Some of the requests control parameters that are also dependent on the content of the incoming MPEG data stream. In general, the behavior of the MPEG decoder is primarily controlled by the incoming bit stream. Parameters set using MPEG Decoder Control requests retain their setting, even if that setting is not applicable to the current incoming bit stream. As an example, consider a decoder that is receiving a stream containing two independent stereo channel pairs. In this case, the Select Second Stereo Control can be enabled so that the second stereo channel is reproduced over the Left and Right channel. If the incoming stream is now switched to a full 5.1 encoded stream, the Select Second Stereo Control has no more influence and the decoder overrides its setting and produces full 5.1 sound. However, if the incoming stream switches back to the previous format, the Select Second Stereo Control becomes active again and resumes its previous setting so that the second stereo channel is reproduced again over the Left and Right channel.

5.2.6.3.1.1 Dual Channel Control

The Dual Channel Control is used to select which of the two available channels in the MPEG-1 base stream is actually retrieved and reproduced over the Left and Right output channels. If this Control is addressed on a decoder that does not implement Dual Channel Control (D4 = 0b0 in the **bmMPEGCapabilities** field of the MPEG format-specific descriptor), the control pipe must indicate a stall.

The Dual Channel Control must have only the CUR attribute. The value of the Dual Channel Control CUR attribute must be either TRUE or FALSE. When FALSE, Channel I is selected, and when TRUE, Channel II is selected

The Control Selector field must be set to MD_DUAL_CHANNEL_CONTROL and the Channel Number field must be set to zero (master Control).

The parameter block for this Control request uses Layout 1 (See Section 5.2.3.1, “Layout 1 Parameter Block.”)

5.2.6.3.1.2 Second Stereo Control

The Second Stereo Control is used to select the second stereo channel pair that can be encoded in an MPEG-2 stream instead of the multi-channel stereophonic information (3/2). If this Control is addressed on a decoder that does not implement Second Stereo support (D5 = 0b0 in the **bmMPEGCapabilities** field of the MPEG format-specific descriptor), the control pipe must indicate a stall.

The Second Stereo Control must have only the CUR attribute. The value of the Second Stereo Control CUR attribute must be either TRUE or FALSE. When FALSE, the main stereo channel pair is selected; when TRUE, the second stereo channel pair is selected.

The Control Selector field must be set to MD_SECOND_STEREO_CONTROL and the Channel Number field must be set to zero (master Control).

The parameter block for this Control request uses Layout 1 (See Section 5.2.3.1, “Layout 1 Parameter Block.”)

5.2.6.3.1.3 Multilingual Control

The Multilingual Control is used to select the multilingual channel actually retrieved from the MPEG stream. If this Control is addressed on a decoder that does not implement multilingual support (D9..8 = 0b00 in the **bmMPEGCapabilities** field of the MPEG format-specific descriptor), the control pipe must indicate a stall.

The Multilingual Control must have only the CUR attribute. The valid range is from zero (0x00) to seven (0x07). The actual range depends on the incoming MPEG stream. It may contain only a limited number of multilingual channels (less than seven).

The Control Selector field must be set to MD_MULTILINGUAL_CONTROL and the Channel Number field must be set to zero (master Control).

The parameter block for this Control request uses Layout 1 (See Section 5.2.3.1, “Layout 1 Parameter Block.”)

5.2.6.3.1.4 Dynamic Range Control

The Dynamic Range Control (DRC) is used to enable or disable the Dynamic Range Control functionality of the decoder. If the decoder does not support Dynamic Range control (D5..4 = 0b00 in the **bmMPEGFeatures** field of the MPEG format-specific descriptor), the control pipe must indicate a stall when receiving this request.

The Dynamic Range Control must have only the CUR attribute. The value of the DRC CUR attribute must be either TRUE or FALSE. TRUE means that the MPEG decoder is using the Dynamic Range control words (possibly with additional scaling) contained in the MPEG bit stream to control the audio dynamic range. FALSE means the control words are being ignored, and the original signal dynamic range is being reproduced.

The Control Selector field must be set to MD_DYN_RANGE_CONTROL and the Channel Number field must be set to zero (master Control).

The parameter block for this Control request uses Layout 1 (See Section 5.2.3.1, “Layout 1 Parameter Block.”)

5.2.6.3.1.5 Scaling Control

The Scaling Control is used to manipulate the single scaling coefficient used by MPEG decoders that implement a common boost/cut scaling value for Dynamic Range Control (D5..4 = 0b10 in the **bmMPEGFeatures** field of the MPEG format-specific descriptor). If this Control is addressed on a non-'10' decoder, the control pipe must indicate a stall.

The Scaling Control must support the CUR and RANGE(MIN, MAX, RES) attributes. The valid range for the CUR, MIN, MAX, and RES attributes is from zero (0x00) to 255/256 (0xFF).

The Control Selector field must be set to MD_SCALING_CONTROL and the Channel Number field must be set to zero (master Control).

The parameter block for this Control request uses Layout 1 (See Section 5.2.3.1, "Layout 1 Parameter Block.")

5.2.6.3.1.6 High/Low Scaling Control

The High/Low Scaling Control is used to manipulate the two scaling coefficients used by MPEG decoders that implement an independent boost and cut scaling value for Dynamic Range Control (D5..4 = 0b11 in the **bmMPEGFeatures** field of the MPEG Decoder descriptor). If this Control is addressed on a non-'11' decoder, the control pipe must indicate a stall.

The High/Low Scaling Control must support the CUR and RANGE(MIN, MAX, RES) attributes. However, if the RANGE attribute is supported, it must do so for both **bCUR_Lo** and **bCUR_Hi** and the MIN, MAX and RES settings must be the same for both. The valid range for the CUR, MIN, MAX, and RES attributes is from zero (0x00) to 255/256 (0xFF). The **bCUR_Lo** value is used by the MPEG decoder to scale the Dynamic Range control words that apply a gain increase (for low sound levels). The **bCUR_Hi** value is used by the MPEG decoder to scale the Dynamic Range control words that apply a gain reduction (for high level sounds).

The Control Selector field must be set to MD_HILO_SCALING_CONTROL and the Channel Number field must be set to zero (master Control).

The parameter block for the CUR attribute of the High/Low Scaling Control is as follows:

Table 5-15: High/Low Scaling Control CUR Parameter Block

wLength		2		
Offset	**Field**	**Size**	**Value**	**Description**
0	bCUR_Lo	1	Number	The setting for the CUR attribute of the low level Scaling Control.
1	bCUR_Hi	1	Number	The setting for the CUR attribute of the high level Scaling Control.

The parameter block for the RANGE attribute of the High/Low Scaling Control is as follows:

Table 5-16: High/Low Scaling Control RANGE Parameter Block

wLength		2+3*n		
Offset	**Field**	**Size**	**Value**	**Description**
0	wNumSubRanges	2	Number	The number of subranges of the High/Low Scaling Control: n

wLength		2+3*n		
Offset	**Field**	**Size**	**Value**	**Description**
2	bMIN(1)	1	Number	The setting for the MIN attribute of the first subrange of the High/Low Scaling Control
3	bMAX(1)	1	Number	The setting for the MAX attribute of the first subrange of the High/Low Scaling Control
4	bRES(1)	1	Number	The setting for the RES attribute of the first subrange of the High/Low Scaling Control
…	…	…	…	…
2+3*(n-1)	bMIN(n)	1	Number	The setting for the MIN attribute of the last subrange of the High/Low Scaling Control
3+3*(n-1)	bMAX(n)	1	Number	The setting for the MAX attribute of the last subrange of the High/Low Scaling Control
4+3*(n-1)	bRES(n)	1	Number	The setting for the RES attribute of the last subrange of the High/Low Scaling Control

5.2.6.3.1.7 Underflow Control

See Section5.2.4.4, “Underflow Control” for a detailed description. XX must be replaced by MD.

5.2.6.3.1.8 Overflow Control

See Section 5.2.4.5, “Overflow Control” for a detailed description. XX must be replaced by MD.

5.2.6.3.1.9 Decoder Error Control

See Section 5.2.4.7, “Decoder Error Control” for a detailed description. XX must be replaced by MD.

5.2.6.3.2 AC-3 Decoder Control Request

This request is used to manipulate the AC-3 Decoder Controls inside an AudioStreaming interface of the audio function. The exact layout of the request is defined in Section 5.2.2, “Control Request Layout.”

The following paragraphs present a detailed description of all possible AC-3 Decoder Controls an AudioStreaming interface can incorporate. For each Control, 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.17.13.2, “AC-3 Decoder Control Selectors.”

5.2.6.3.2.1 Mode Control

The Mode Control is used to change the compression mode of the AC-3 decoder in the AudioStreaming interface. A Mode Control must only support the CUR attribute. The valid range for the CUR attribute is described through the **bmComprFeatures** field of the AC-3 Decoder descriptor. Bits D3..0 describe which compression modes the AC-3 decoder supports. Valid values are:

- 0: RF mode
- 1: Line mode
- 2: Custom0 mode
- 3: Custom1 mode

If the Mode Control request specifies an unsupported mode, the control pipe must indicate a stall.

The Control Selector field must be set to AD_MODE_CONTROL and the Channel Number field must be set to zero (master Control).

The parameter block for this Control request uses Layout 1 (See Section 5.2.3.1, “Layout 1 Parameter Block.”)

5.2.6.3.2.2 Dynamic Range Control

The Dynamic Range Control (DRC) is used to enable or disable the Dynamic Range Control functionality of the decoder. The Dynamic Range Control must have only the CUR attribute. The value of the Dynamic Range Control CUR attribute must be either TRUE or FALSE. TRUE means that the AC-3 decoder is using the Dynamic Range control words (possibly with additional scaling) contained in the AC-3 bit stream to control the audio dynamic range. FALSE means the control words are being ignored and the original signal dynamic range is being reproduced.

The Control Selector field must be set to AD_DYN_RANGE_CONTROL and the Channel Number field must be set to zero (master Control).

The parameter block for this Control request uses Layout 1 (See Section 5.2.3.1, “Layout 1 Parameter Block.”)

5.2.6.3.2.3 Scaling Control

The Scaling Control is used to manipulate the single scaling coefficient used by AC-3 decoders that implement a common boost/cut scaling value for Dynamic Range Control. (D5..4 = 0b10 in the **bmAC3Features** field of the AC-3 Decoder descriptor.) If this Control is addressed on a non-‘10’ decoder, the control pipe must indicate a stall.

The Scaling Control must support the CUR and RANGE(MIN, MAX, RES) attributes. The valid range for the CUR, MIN, MAX, and RES attributes is from zero (0x00) to 255/256 (0xFF).

The Control Selector field must be set to AD_SCALING_CONTROL and the Channel Number field must be set to zero (master Control).

The parameter block for this Control request uses Layout 1 (See Section 5.2.3.1, “Layout 1 Parameter Block.”)

5.2.6.3.2.4 High/Low Scaling Control

The High/Low Scaling Control is used to manipulate the two scaling coefficients used by AC-3 decoders that implement an independent boost and cut scaling value for Dynamic Range Control. (D5..4 = 0b11 in the **bmAC3Features** field of the AC-3 Decoder descriptor.) If this Control is addressed on a non-‘11’ decoder, the control pipe must indicate a stall.

The High/Low Scaling Control must support the CUR and RANGE(MIN, MAX, RES) attributes. The valid range for the CUR, MIN, MAX, and RES attributes is from zero (0x00) to 255/256 (0xFF). The **bCUR_Lo** value is used by the AC-3 decoder to scale the Dynamic Range control words which apply a gain increase (for low sound levels). The **bCUR_Hi** value is used by the AC-3 decoder to scale the Dynamic Range control words which apply a gain reduction (for high level sounds).

The Control Selector field must be set to AD_HILO_SCALING_CONTROL and the Channel Number field must be set to zero (master Control).

The parameter block for the CUR attribute of the High/Low Scaling Control is as follows:

Table 5-17: High/Low Scaling Control CUR Parameter Block

wLength		2		
Offset	**Field**	**Size**	**Value**	**Description**
0	bCUR_Lo	1	Number	The setting for the CUR attribute of the low level Scaling Control.
1	bCUR_Hi	1	Number	The setting for the CUR attribute of the high level Scaling Control.

The parameter block for the RANGE attribute of the High/Low Scaling Control is as follows:

Table 5-18: High/Low Scaling Control RANGE Parameter Block

wLength		2+3*n		
Offset	**Field**	**Size**	**Value**	**Description**
0	wNumSubRanges	2	Number	The number of subranges of the High/Low Scaling Control: n
2	bMIN(1)	1	Number	The setting for the MIN attribute of the first subrange of the High/Low Scaling Control
3	bMAX(1)	1	Number	The setting for the MAX attribute of the first subrange of the High/Low Scaling Control
4	bRES(1)	1	Number	The setting for the RES attribute of the first subrange of the High/Low Scaling Control
…	…	…	…	…
2+3*(n-1)	bMIN(n)	1	Number	The setting for the MIN attribute of the last subrange of the High/Low Scaling Control
3+3*(n-1)	bMAX(n)	1	Number	The setting for the MAX attribute of the last subrange of the High/Low Scaling Control
4+3*(n-1)	bRES(n)	1	Number	The setting for the RES attribute of the last subrange of the High/Low Scaling Control

5.2.6.3.2.5 Underflow Control

See Section5.2.4.4, “Underflow Control” for a detailed description. XX must be replaced by AD.

5.2.6.3.2.6 Overflow Control

See Section 5.2.4.5, “Overflow Control” for a detailed description. XX must be replaced by AD.

5.2.6.3.2.7 Decoder Error Control

See Section 5.2.4.7, “Decoder Error Control” for a detailed description. XX must be replaced by AD.

5.2.6.3.3 WMA Decoder Control Request

This request is used to manipulate the WMA Decoder Controls inside an AudioStreaming interface of the audio function. The exact layout of the request is defined in Section 5.2.2, “Control Request Layout.”

The following paragraphs present a detailed description of all possible WMA Decoder Controls an AudioStreaming interface can incorporate. For each Control, 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.17.13.3, “WMA Decoder Control Selectors.”

5.2.6.3.3.1 Underflow Control

See Section5.2.4.4, “Underflow Control” for a detailed description. XX must be replaced by WD.

5.2.6.3.3.2 Overflow Control

See Section 5.2.4.5, “Overflow Control” for a detailed description. XX must be replaced by WD.

5.2.6.3.3.3 Decoder Error Control

See Section 5.2.4.7, “Decoder Error Control” for a detailed description. XX must be replaced by WD.

5.2.6.3.4 DTS Decoder Control Request

This request is used to manipulate the DTS Decoder Controls inside an AudioStreaming interface of the audio function. The exact layout of the request is defined in Section 5.2.2, “Control Request Layout.”

The following paragraphs present a detailed description of all possible DTS Decoder Controls an AudioStreaming interface can incorporate. For each Control, 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.17.13.4, “DTS Decoder Control Selectors.”

5.2.6.3.4.1 Underflow Control

See Section5.2.4.4, “Underflow Control” for a detailed description. XX must be replaced by DD.

5.2.6.3.4.2 Overflow Control

See Section 5.2.4.5, “Overflow Control” for a detailed description. XX must be replaced by DD.

5.2.6.3.4.3 Decoder Error Control

See Section 5.2.4.7, “Decoder Error Control” for a detailed description. XX must be replaced by DD.

5.2.6.4 Endpoint Control Request

This request is used to manipulate the Controls inside an AudioStreaming endpoint of the audio function. The exact layout of the request is defined in Section 5.2.2, “Control Request Layout.”

The following paragraphs present a detailed description of all possible Controls an Endpoint can incorporate. For each Control, 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 AppendixA.17.14, “Endpoint Control Selectors.”

5.2.6.4.1 Pitch Control

The Pitch Control enables or disables the ability of an adaptive endpoint to dynamically track its sampling frequency. The Control is necessary because the clock recovery circuitry must be informed whether it should allow for relatively large swings in the sampling frequency. A Pitch Control must have only the CUR attribute. The value of a Pitch Control CUR attribute must be either TRUE or FALSE.

The Control Selector field must be set to EP_PITCH_CONTROL and the Channel Number field must be set to zero (master Control).

The parameter block for this Control request uses Layout 1 (See Section 5.2.3.1, “Layout 1 Parameter Block.”)

5.2.6.4.2 Data Overrun Control

The Data Overrun Control is used to indicate the occurence of a data overrun (buffer overflow) condition within an Endpoint *since the last Get Data Overrun request*. This Control only supports the Get request (read-only). A Data Overrun Control must have only the CUR attribute. The value of a Data Overrun Control CUR attribute must be either TRUE (overrun condition occured) or FALSE (normal).

The Control Selector field must be set to EP_DATA_OVERRUN_CONTROL and the Channel Number field must be set to zero (master Control).

The parameter block for this Control request uses Layout 1 (See Section 5.2.3.1, “Layout 1 Parameter Block.”)

5.2.6.4.3 Data Underrun Control

The Data Underrun Control is used to indicate the occurence of a data underrun (buffer underflow) condition within an Endpoint *since the last Get Data Underrun request*. This Control only supports the Get request (read-only). A Data Underrun Control must have only the CUR attribute. The value of a Data Underrun Control CUR attribute must be either TRUE (underrun condition occured) or FALSE (normal).

The Control Selector field must be set to EP_DATA_UNDERRUN_CONTROL and the Channel Number field must be set to zero (master Control).

The parameter block for this Control request uses Layout 1 (See Section 5.2.3.1, “Layout 1 Parameter Block.”)

5.2.7 Additional Requests

5.2.7.1 Memory Requests

The Host can interact with an addressable Entity (Clock Entity, Terminal, Unit, Interface or endpoint) within the audio function in a very generic way. The Entity presents a memory space to the Host whose layout depends on the implementation. The Memory request provides full access to this memory space.

This request is used to upload or download a parameter block into a particular Entity of the audio function.

Table 5-19: Memory Request Values

bmRequest Type	bRequest	wValue	wIndex	wLength	Data
00100001B 10100001B	MEM	Offset	Entity ID and Interface	Length of parameter block	Parameter block
00100010B 10100010B			Endpoint		

The **bRequest** field indicates that the MEM attribute of the Entity is addressed.

The **wValue** field specifies a zero-based offset value that can be used to access only parts of the Entity's memory space.

The **wIndex** field specifies the interface or endpoint to be addressed in the low byte and the Entity ID (Clock Entity ID, Unit ID or Terminal ID) or zero in the high byte. In case an interface is addressed, the virtual Entity 'interface' can be addressed by specifying zero in the high byte. The values in **wIndex** must be appropriate to the recipient. Only existing Entities in the audio function can be addressed and only appropriate interface or endpoint numbers may be used. If the request specifies an unknown or non-Entity ID or an unknown interface or endpoint number, the control pipe must indicate a stall.

The layout of the parameter block is implementation dependent. A device is required to reevaluate its memory space at the end of each Set Memory request.

6 Interrupts

Interrupts are used as a means to inform the Host that a change has occurred in the current state of the audio function. This specification currently defines two different types of interrupts:

- Memory Change: Some internal Entity's memory location has been updated. Host software can be notified so that the appropriate action can be taken.
- Control Change: Some addressable Control inside the audio function changed one or more of its attribute values.

The Audio Controls inside a Clock Entity, Unit, or Terminal can be the source of an interrupt. Likewise, any addressable Control inside the AudioControl interface or any of the AudioStreaming interfaces can generate an interrupt. Finally, all addressable Controls related to an audio endpoint can be the cause of an interrupt.

A change of state in the audio function is most often caused by a certain event that takes place. An event can either be user-initiated or device-initiated. User-initiated jack insertion or removal is a typical example of a user-initiated event. The Host could then 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 could be the following: An external device (e.g. an A/V receiver) could switch 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 might need to be reconfigured (e.g. to start the AC-3 decoding process), maybe causing all 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 Data Message

The actual type of interrupt (Memory Change or Control Change) and its originator is conveyed to the Host through the interrupt data message that is sent over the interrupt endpoint. It is then the responsibility of the Host to query the audio function for more detailed information about the cause of the interrupt through a Get Memory request or one of the Get Control requests as defined in Section 5.2, "Class-Specific Requests".

Interrupts are considered to be of the 'edge-triggered' type, meaning that an interrupt is generated whenever an event occurs, but there is no specific action required from the Host to clear the interrupt condition. When the Host issues a Get request in response to the interrupt, the most current value of the addressed Control's attribute will be returned.

The interrupt data message is always 6 bytes in length. The first **bInfo** field is required for all interrupt data messages. It contains information in D0 indicating whether this is a vendor-specific interrupt (D0 = 0b1) or a class-specific interrupt (D0 = 0b0). Bit D1 indicates whether the interrupt originated from an interface (D1 = 0b0) or an endpoint (D1 = 0b1). Bits D7..2 of the **bInfo** field are reserved for future extensions and must be set to zero. For vendor-specific interrupts, the layout of the remainder of the interrupt message is undefined. For class-specific interrupts, the layout is defined as follows.

When the interrupt originates from an Entity in an interface (D1 = 0b0 in the **bInfo** field), the **wIndex** field specifies the interface in the low byte and the Entity ID (Clock Entity ID, Unit ID, Terminal ID, Encoder ID, or Decoder ID). For indicating the interface itself, an Entity ID of zero must be specified in the high byte.

When the interrupt originates from an endpoint (D1 = 0b1 in the **bInfo** field), the **wIndex** field specifies the endpoint to be addressed in the low byte and zero in the high byte.

The **wValue** field interpretation is qualified by the value in the **wIndex** field. The layout of the **wValue** field changes depending on the addressed Entity. The **wValue** field follows exactly the same rules as outlined in Section 5, "Requests" for each of the supported Get Control requests. The **wValue** field returns

the Control Selector (CS) in the high byte and the Channel Number (CN) in the low byte. The Control Selector and the Channel Number (CN) together indicate exactly which Control generated the interrupt. If a Control is channel-independent, then the Control is considered to be a master Control and the virtual channel zero is returned to indicate it (CN = 0).

There are two exceptions to the above. The first is when a Mixer Unit Control request returns CS = MU_MIXER_CONTROL in the high byte. Then the Mixer Control Number (MCN) is returned in the low byte. The second is the Memory request where the **wValue** field specifies a zero-based offset value that indicates the address of the location in the Entity's memory space that generated the interrupt. If the offset value is zero, this indicates that multiple memory locations might have changed and the entire memory space needs to be examined.

The **bAttribute** field contains a constant, identifying which attribute of the addressed Control or Entity is causing the interrupt. Possible attributes are:

- Current setting attribute (CUR)
- Range attribute (RANGE)
- Memory space attribute (MEM)

When there are no interrupts pending, the interrupt endpoint must NAK when polled.

The following table specifies the format of the interrrupt message when D0 = 0b0:

Table 6-1: Interrupt Data Message Format

Offset	Field	Size	Value	Description
0	bInfo	1	Bitmap	D0: Vendor-specific. D1: Interface or Endpoint D7..2: Reserved for future extensions. Must be set to 0.
1	bAttribute	1	Constant	The attribute that caused the interrupt
2	wValue	2	Number	CS in the high byte and CN or MCN in the low byte.
4	wIndex	2	Number	Entity ID or zero in the high byte and Interface or Endpoint in the low byte.

6.2 Interrupt Sources

Any Control within an addressable Entity of the audio function can be the source of an interrupt. The interrupt message contains enough information to determine exactly which Control caused the interrupt. The Host can then issue the normal Control requests to further qualify the cause of the interrupt.

Appendix A. Audio Device Class Codes

A.1 Audio Function Class Code

Table A-1: Audio Function Class Code

Audio Function Class Code	Value
AUDIO_FUNCTION	AUDIO

A.2 Audio Function Subclass Codes

Table A-2: Audio Function Subclass Codes

Audio Function Subclass Code	Value
FUNCTION_SUBCLASS_UNDEFINED	0x00

A.3 Audio Function Protocol Codes

Table A-3: Audio Function Protocol Codes

Audio Function Protocol Code	Value
FUNCTION_PROTOCOL_UNDEFINED	0x00
AF_VERSION_02_00	IP_VERSION_02_00

A.4 Audio Interface Class Code

Table A-4: Audio Interface Class Code

Audio Interface Class Code	Value
AUDIO	0x01

A.5 Audio Interface Subclass Codes

Table A-5: Audio Interface Subclass Codes

Audio Interface Subclass Code	Value
INTERFACE_SUBCLASS_UNDEFINED	0x00
AUDIOCONTROL	0x01
AUDIOSTREAMING	0x02
MIDISTREAMING	0x03

A.6 Audio Interface Protocol Codes

Table A-6: Audio Interface Protocol Codes

Audio Interface Protocol Code	Value
INTERFACE_PROTOCOL_UNDEFINED	0x00
IP_VERSION_02_00	0x20

A.7 Audio Function Category Codes

Table A-7: Audio Function Category Codes

Audio Function Subclass Code	Value
FUNCTION_SUBCLASS_UNDEFINED	0x00
DESKTOP_SPEAKER	0x01
HOME_THEATER	0x02
MICROPHONE	0x03
HEADSET	0x04
TELEPHONE	0x05
CONVERTER	0x06
VOICE/SOUND_RECORDER	0x07
I/O_BOX	0x08
MUSICAL_INSTRUMENT	0x09
PRO-AUDIO	0x0A
AUDIO/VIDEO	0x0B
CONTROL_PANEL	0x0C
OTHER	0xFF

A.8 Audio Class-Specific Descriptor Types

Table A-8: Audio Class-specific Descriptor Types

Descriptor Type	Value
CS_UNDEFINED	0x20
CS_DEVICE	0x21
CS_CONFIGURATION	0x22

Descriptor Type	Value
CS_STRING	0x23
CS_INTERFACE	0x24
CS_ENDPOINT	0x25

A.9 Audio Class-Specific AC Interface Descriptor Subtypes

Table A-9: Audio Class-Specific AC Interface Descriptor Subtypes

Descriptor Subtype	Value
AC_DESCRIPTOR_UNDEFINED	0x00
HEADER	0x01
INPUT_TERMINAL	0x02
OUTPUT_TERMINAL	0x03
MIXER_UNIT	0x04
SELECTOR_UNIT	0x05
FEATURE_UNIT	0x06
EFFECT_UNIT	0x07
PROCESSING_UNIT	0x08
EXTENSION_UNIT	0x09
CLOCK_SOURCE	0x0A
CLOCK_SELECTOR	0x0B
CLOCK_MULTIPLIER	0x0C
SAMPLE_RATE_CONVERTER	0x0D

A.10 Audio Class-Specific AS Interface Descriptor Subtypes

Table A-10: Audio Class-Specific AS Interface Descriptor Subtypes

Descriptor Subtype	Value
AS_DESCRIPTOR_UNDEFINED	0x00
AS_GENERAL	0x01
FORMAT_TYPE	0x02
ENCODER	0x03

Descriptor Subtype	Value
DECODER	0x04

A.11 Effect Unit Effect Types

Table A-11: Effect Unit Effect Types

wEffectType	Value
EFFECT_UNDEFINED	0x00
PARAM_EQ_SECTION_EFFECT	0x01
REVERBERATION_EFFECT	0x02
MOD_DELAY_EFFECT	0x03
DYN_RANGE_COMP_EFFECT	0x04

A.12 Processing Unit Process Types

Table A-12: Processing Unit Process Types

wProcessType	Value
PROCESS_UNDEFINED	0x00
UP/DOWNMIX_PROCESS	0x01
DOLBY_PROLOGIC_PROCESS	0x02
STEREO_EXTENDER_PROCESS	0x03

A.13 Audio Class-Specific Endpoint Descriptor Subtypes

Table A-13: Audio Class-Specific Endpoint Descriptor Subtypes

Descriptor Subtype	Value
DESCRIPTOR_UNDEFINED	0x00
EP_GENERAL	0x01

A.14 Audio Class-Specific Request Codes

Table A-14: Audio Class-Specific Request Codes

Class-Specific Request Code	Value
REQUEST_CODE_UNDEFINED	0x00
CUR	0x01
RANGE	0x02

Class-Specific Request Code	Value
MEM	0x03

A.15 Encoder Type Codes

Table A-15: Encoder Type Codes

Encoder Type Code	Value
ENCODER_UNDEFINED	0x00
OTHER_ENCODER	0x01
MPEG_ENCODER	0x02
AC-3_ENCODER	0x03
WMA_ENCODER	0x04
DTS_ENCODER	0x05

A.16 Decoder Type Codes

Table A-16: Decoder Type Codes

Decoder Type Code	Value
DECODER_UNDEFINED	0x00
OTHER_DECODER	0x01
MPEG_DECODER	0x02
AC-3_DECODER	0x03
WMA_DECODER	0x04
DTS_DECODER	0x05

A.17 Control Selector Codes

A.17.1 Clock Source Control Selectors

Table A-17: Clock Source Control Selectors

Control Selector	Value
CS_CONTROL_UNDEFINED	0x00
CS_SAM_FREQ_CONTROL	0x01
CS_CLOCK_VALID_CONTROL	0x02

A.17.2 Clock Selector Control Selectors

Table A-18: Clock Selector Control Selectors

Control Selector	Value
CX_CONTROL_UNDEFINED	0x00
CX_CLOCK_SELECTOR_CONTROL	0x01

A.17.3 Clock Multiplier Control Selectors

Table A-19: Clock Multiplier Control Selectors

Control Selector	Value
CM_CONTROL_UNDEFINED	0x00
CM_NUMERATOR_CONTROL	0x01
CM_DENOMINATOR_CONTROL	0x02

A.17.4 Terminal Control Selectors

Table A-20: Terminal Control Selectors

Control Selector	Value
TE_CONTROL_UNDEFINED	0x00
TE_COPY_PROTECT_CONTROL	0x01
TE_CONNECTOR_CONTROL	0x02
TE_OVERLOAD_CONTROL	0x03
TE_CLUSTER_CONTROL	0x04
TE_UNDERFLOW_CONTROL	0x05
TE_OVERFLOW_CONTROL	0x06
TE_LATENCY_CONTROL	0x07

A.17.5 Mixer Control Selectors

Table A-21: Mixer Control Selectors

Control Selector	Value
MU_CONTROL_UNDEFINED	0x00
MU_MIXER_CONTROL	0x01
MU_CLUSTER_CONTROL	0x02

Control Selector	Value
MU_UNDERFLOW_CONTROL	0x03
MU_OVERFLOW_CONTROL	0x04
MU_LATENCY_CONTROL	0x05

A.17.6 Selector Control Selectors

Table A-22: Selector Control Selectors

Control Selector	Value
SU_CONTROL_UNDEFINED	0x00
SU_SELECTOR_CONTROL	0x01
SU_LATENCY_CONTROL	0x02

A.17.7 Feature Unit Control Selectors

Table A-23: Feature Unit Control Selectors

Control Selector	Value
FU_CONTROL_UNDEFINED	0x00
FU_MUTE_CONTROL	0x01
FU_VOLUME_CONTROL	0x02
FU_BASS_CONTROL	0x03
FU_MID_CONTROL	0x04
FU_TREBLE_CONTROL	0x05
FU_GRAPHIC_EQUALIZER_CONTROL	0x06
FU_AUTOMATIC_GAIN_CONTROL	0x07
FU_DELAY_CONTROL	0x08
FU_BASS_BOOST_CONTROL	0x09
FU_LOUDNESS_CONTROL	0x0A
FU_INPUT_GAIN_CONTROL	0x0B
FU_INPUT_GAIN_PAD_CONTROL	0x0C
FU_PHASE_INVERTER_CONTROL	0x0D
FU_UNDERFLOW_CONTROL	0x0E

Control Selector	Value
FU_OVERFLOW_CONTROL	0x0F
FU_LATENCY_CONTROL	0x10

A.17.8 Effect Unit Control Selectors

A.17.8.1 Parametric Equalizer Section Effect Unit Control Selectors

Table A-24: Reverberation Effect Unit Control Selectors

Control Selector	Value
PE_CONTROL_UNDEFINED	0x00
PE_ENABLE_CONTROL	0x01
PE_CENTERFREQ_CONTROL	0x02
PE_QFACTOR_CONTROL	0x03
PE_GAIN_CONTROL	0x04
PE_UNDERFLOW_CONTROL	0x05
PE_OVERFLOW_CONTROL	0x06
PE_LATENCY_CONTROL	0x07

A.17.8.2 Reverberation Effect Unit Control Selectors

Table A-25: Reverberation Effect Unit Control Selectors

Control Selector	Value
RV_CONTROL_UNDEFINED	0x00
RV_ENABLE_CONTROL	0x01
RV_TYPE_CONTROL	0x02
RV_LEVEL_CONTROL	0x03
RV_TIME_CONTROL	0x04
RV_FEEDBACK_CONTROL	0x05
RV_PREDELAY_CONTROL	0x06
RV_DENSITY_CONTROL	0x07
RV_HIFREQ_ROLLOFF_CONTROL	0x08

Control Selector	Value
RV_UNDERFLOW_CONTROL	0x09
RV_OVERFLOW_CONTROL	0x0A
RV_LATENCY_CONTROL	0x0B

A.17.8.3 Modulation Delay Effect Unit Control Selectors

Table A-26: Modulation Delay Effect Unit Control Selectors

Control Selector	Value
MD_CONTROL_UNDEFINED	0x00
MD_ENABLE_CONTROL	0x01
MD_BALANCE_CONTROL	0x02
MD_RATE_CONTROL	0x03
MD_DEPTH_CONTROL	0x04
MD_TIME_CONTROL	0x05
MD_FEEDBACK_CONTROL	0x06
MD_UNDERFLOW_CONTROL	0x07
MD_OVERFLOW_CONTROL	0x08
MD_LATENCY_CONTROL	0x09

A.17.8.4 Dynamic Range Compressor Effect Unit Control Selectors

Table A-27: Dynamic Range Compressor Effect Unit Control Selectors

Control Selector	Value
DR_CONTROL_UNDEFINED	0x00
DR_ENABLE_CONTROL	0x01
DR_COMPRESSION_RATE_CONTROL	0x02
DR_MAXAMPL_CONTROL	0x03
DR_THRESHOLD_CONTROL	0x04
DR_ATTACK_TIME_CONTROL	0x05
DR_RELEASE_TIME_CONTROL	0x06
DR_UNDERFLOW_CONTROL	0x07

Control Selector	Value
DR_OVERFLOW_CONTROL	0x08
DR_LATENCY_CONTROL	0x09

A.17.9 Processing Unit Control Selectors

A.17.9.1 Up/Down-mix Processing Unit Control Selectors

Table A-28: Up/Down-mix Processing Unit Control Selectors

Control Selector	Value
UD_CONTROL_UNDEFINED	0x00
UD_ENABLE_CONTROL	0x01
UD_MODE_SELECT_CONTROL	0x02
UD_CLUSTER_CONTROL	0x03
UD_UNDERFLOW_CONTROL	0x04
UD_OVERFLOW_CONTROL	0x05
UD_LATENCY_CONTROL	0x06

A.17.9.2 Dolby Prologic™ Processing Unit Control Selectors

Table A-29: Dolby Prologic Processing Unit Control Selectors

Control Selector	Value
DP_CONTROL_UNDEFINED	0x00
DP_ENABLE_CONTROL	0x01
DP_MODE_SELECT_CONTROL	0x02
DP_CLUSTER_CONTROL	0x03
DP_UNDERFLOW_CONTROL	0x04
DP_OVERFLOW_CONTROL	0x05
DP_LATENCY_CONTROL	0x06

A.17.9.3 Stereo Extender Processing Unit Control Selectors

Table A-30: Stereo Extender Processing Unit Control Selectors

Control Selector	Value
ST_EXT_CONTROL_UNDEFINED	0x00
ST_EXT_ENABLE_CONTROL	0x01
ST_EXT_WIDTH_CONTROL	0x02
ST_EXT_UNDERFLOW_CONTROL	0x03
ST_EXT_OVERFLOW_CONTROL	0x04
ST_EXT_LATENCY_CONTROL	0x05

A.17.10 Extension Unit Control Selectors

Table A-31: Extension Unit Control Selectors

Control Selector	Value
XU_CONTROL_UNDEFINED	0x00
XU_ENABLE_CONTROL	0x01
XU_CLUSTER_CONTROL	0x02
XU_UNDERFLOW_CONTROL	0x03
XU_OVERFLOW_CONTROL	0x04
XU_LATENCY_CONTROL	0x05

A.17.11 AudioStreaming Interface Control Selectors

Table A-32: AudioStreaming Interface Control Selectors

Control Selector	Value
AS_CONTROL_UNDEFINED	0x00
AS_ACT_ALT_SETTING_CONTROL	0x01
AS_VAL_ALT_SETTINGS_CONTROL	0x02
AS_AUDIO_DATA_FORMAT_CONTROL	0x03

A.17.12 Encoder Control Selectors

Table A-33: Encoder Control Selectors

Control Selector	Value
EN_CONTROL_UNDEFINED	0x00
EN_BIT_RATE_CONTROL	0x01
EN_QUALITY_CONTROL	0x02
EN_VBR_CONTROL	0x03
EN_TYPE_CONTROL	0x04
EN_UNDERFLOW_CONTROL	0x05
EN_OVERFLOW_CONTROL	0x06
EN_ENCODER_ERROR_CONTROL	0x07
EN_PARAM1_CONTROL	0x08
EN_PARAM2_CONTROL	0x09
EN_PARAM3_CONTROL	0x0A
EN_PARAM4_CONTROL	0x0B
EN_PARAM5_CONTROL	0x0C
EN_PARAM6_CONTROL	0x0D
EN_PARAM7_CONTROL	0x0E
EN_PARAM8_CONTROL	0x0F

A.17.13 Decoder Control Selectors

A.17.13.1 MPEG Decoder Control Selectors

Table A-34: MPEG Decoder Control Selectors

Control Selector	Value
MD_CONTROL_UNDEFINED	0x00
MD_DUAL_CHANNEL_CONTROL	0x01
MD_SECOND_STEREO_CONTROL	0x02
MD_MULTILINGUAL_CONTROL	0x03
MD_DYN_RANGE_CONTROL	0x04

Control Selector	Value
MD_SCALING_CONTROL	0x05
MD_HILO_SCALING_CONTROL	0x06
MD_UNDERFLOW_CONTROL	0x07
MD_OVERFLOW_CONTROL	0x08
MD_DECODER_ERROR_CONTROL	0x09

A.17.13.2 AC-3 Decoder Control Selectors

Table A-35: AC-3 Decoder Control Selectors

Control Selector	Value
AD_CONTROL_UNDEFINED	0x00
AD_MODE_CONTROL	0x01
AD_DYN_RANGE_CONTROL	0x02
AD_SCALING_CONTROL	0x03
AD_HILO_SCALING_CONTROL	0x04
AD_UNDERFLOW_CONTROL	0x05
AD_OVERFLOW_CONTROL	0x06
AD_DECODER_ERROR_CONTROL	0x07

A.17.13.3 WMA Decoder Control Selectors

Table A-36: WMA Decoder Control Selectors

Control Selector	Value
WD_CONTROL_UNDEFINED	0x00
WD_UNDERFLOW_CONTROL	0x01
WD_OVERFLOW_CONTROL	0x02
WD_DECODER_ERROR_CONTROL	0x03

A.17.13.4 DTS Decoder Control Selectors

Table A-37: DTS Decoder Control Selectors

Control Selector	Value
DD_CONTROL_UNDEFINED	0x00

Control Selector	Value
DD_UNDERFLOW_CONTROL	0x01
DD_OVERFLOW_CONTROL	0x02
DD_DECODER_ERROR_CONTROL	0x03

A.17.14 Endpoint Control Selectors

Table A-38: Endpoint Control Selectors

Control Selector	Value
EP_CONTROL_UNDEFINED	0x00
EP_PITCH_CONTROL	0x01
EP_DATA_OVERRUN_CONTROL	0x02
EP_DATA_UNDERRUN_CONTROL	0x03