Universal Serial Bus
Device Class Definition
for
Audio Devices

Release 1.0

March 18, 1998

Scope of This Release

This document is the 1.0 release of this device class definition.

Contributors

Gal Ashour	IBM Corporation
Billy Brackenridge	Microsoft Corporation
Oren Tirosh	Altec Lansing
Craig Todd	Dolby Laboratories
Remy Zimmermann	Logitech
Geert Knapen	Philips ITCL Interleuvenlaan 74-76 B-3001 Leuven-Heverlee BELGIUM Phone: +32 16 390 734 Fax: +32 16 390 600 E-mail: Geert.Knapen@innet.be

Revision History

Revision	Date	Filename	Author	Description
0.1	Aug. 7, 95	Audio01.doc	Geert Knapen	Initial version.
0.2	Aug. 28, 95	Audio01.doc	Geert Knapen	Corrected typos. Attributes field from 8 to 16 bits. Auxiliary channel definition. Important issues added.
0.3	Oct. 9, 95	Audio03.doc	Geert Knapen	Intermediate version.
0.4	Nov. 29, 95	Audio04.doc	Geert Knapen	Change to Audio Function and Interface Property requests. Synch issues updated. Subclass divisions changed.
0.6	Dec. 19, 95	Audio06.doc	Geert Knapen	Listed remarks from last f2f Dec 7-8.
0.8	Dec. 12, 95	Audio08.doc	Geert Knapen	Incorporated changes, discussed at f2f Dec 6 95.
0.8a	Jan. 20, 96	Audio08a.doc	Geert Knapen	Incorporated changes discussed at f2f Jan 18 95. Feedforward/feedback endpoint is now called synch endpoint.
	Feb. 5, 96	usb_au8a.doc		Edited version of Audio08a.doc.
0.8b	June. 5, 96	Audio08b.doc	Geert Knapen	Introduced new mixer concepts etc.
0.8c	Oct. 1, 96	Audio08c.doc	Geert Knapen	Added appropriate descriptors and requests.
0.8d	Dec. 1, 96	Audio08d.doc	Geert Knapen	Included remarks on 0.8c

Revision	Date	Filename	Author	Description
0.8e	Jan. 1, 97	Audio08e.doc	Geert Knapen	Included remarks on 0.8d. Added Dolby Prologic and Up/Down-mix Processing Units.
0.8f	Mar. 1, 97	Audio08f.doc	Geert Knapen	Removed associated interface. Added Set/Get Memory requests for all Entities. Introduced copyright protection, Audio Interface Collections. Added Stereo Widening Processing Unit. Added Reverb Processing Unit. Added Chorus Unit. Added Bass Boost and Loudness Controls.
0.9rc	Apr. 1, 97	Audio09rc.doc	Geert Knapen	Changed Section 5 structure. Removed many request codes. Added requests for Reverb and Chorus. Changed Terminal request structure. Included all remarks from last meeting.
0.9	May 1, 97	Audio09.doc	Geert Knapen	Added wLockDelay and bLockUnits fields to CS endpoint descriptor. Added bit to CS endpoint descriptor to indicate packet size restrictions. Revised endpoint descriptors according to new CCS layout. Added Dynamic Range Compressor PU.
0.9CE	Sep 1, 97	Audio09CE.doc	Geert Knapen	Copy-edited for publication on the web.
0.9a	Oct 1, 97	Audio09a.doc	Geert Knapen	Incorporated RRs
1.0RC	Mar 1, 98	Audio10RC.doc	Geert Knapen	Added examples and cleaned up the formatting.
1.0	Mar 18, 98	Audio10.doc	Geert Knapen	Changed all references to 1.0.

INTELLECTUAL PROPERTY DISCLAIMER

Please send comments via electronic mail to techsup@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*, 1.0 final draft revision (also referred to as the *USB Specification*). In particular, see Section 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/52 Dec. 20, 1995. (available from http://www.atsc.org)
- ANSI/IEEE-754 floating-point standard.
- ISO/IEC 958 International Standard: *Digital Audio Interface and Annexes.*
- ISO/IEC 1937 standard.
- ITU G.711 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 Section 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.
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	**L**ogical 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 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 (voice and sound). PC-based voice telephony is one of the major drivers of USB technology. In addition, the USB has more than enough bandwidth for sound, even high-quality audio. Many applications related to voice telephony, audio playback, and recording can take advantage of the USB.

In principle, a versatile bus specification like the USB provides many ways to propagate and 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 USB. However, provisions have been made so that vendor-specific audio formats and compression schemes can be handled.

3 Functional Characteristics

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 CD-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.

Audio functions are addressed through their audio interfaces. Each audio function has a single AudioControl interface and can have several AudioStreaming and 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 are 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.

Note: All MIDI-related information is grouped in a separate document, *Universal Serial Bus Device Class Definition for MIDIStreaming Interfaces* that is considered part of this specification.

3.1 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 Section A.1, “Audio Interface Class Code.”

3.2 Audio Interface Subclass and Protocol

The Audio Interface class is divided into Subclasses that can be further qualified by the Interface Protocol code. However, at this moment, the Interface Protocol is not used and must be set to 0x00. All audio functions are part of a certain Subclass. The following three Subclasses are currently defined in this specification:

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

The assigned codes can be found in Sections A.2, “Audio Interface Subclass Codes” and A.3, “Audio Interface Protocol Codes” of this specification. All other Subclass codes are unused and reserved except code 0xFF which is by specification reserved for vendor-specific extensions.

3.3 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.

3.3.1 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.

3.3.2 Synchronous

The clock system of synchronous isochronous audio endpoints can be controlled externally through SOF synchronization. Such an endpoint must do one of the following:

- Slave its sample clock to the 1ms SOF tick.
- Control the rate of USB SOF generation so that its data rate becomes automatically locked to SOF.

3.3.3 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.

3.4 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 frames (ms) and is due to the fact that the audio function must buffer at least one frame worth of samples to effectively remove packet jitter within a 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 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 frame *n*) is the first sample of the packet it sends over USB during frame ($n+\delta$). δ is the audio function's internal delay expressed in ms. The same rule applies for an audio sink function. The first sample in the packet, received over USB during frame *n*, must be the first sample that is fully reproduced during 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.

3.5 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.

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. Units are wired together by connecting their I/O Pins according to the required topology.

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. Terminals only have one Input or Output Pin that 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 seven 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
- Output Terminal
- Mixer Unit
- Selector Unit
- Feature Unit
- Processing Unit
- Extension Unit

The ensemble of UDs and TDs provide a full description of the audio function to the Host. 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.

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

Inside a Unit, functionality is further described through audio Controls. A Control typically provides access to a specific audio 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 four attributes:

- Current setting attribute
- 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 (Unit or Terminal) 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.

3.5.1 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 that endpoint and its associated Input Terminal. The class-specific endpoint descriptor contains a field that holds a direct reference to this Input Terminal. The Host needs to use both the 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 endpoint 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. 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 Left, Left Center, Center, Right Center, Right, Right Surround, Left Surround and Low Frequency Enhancement 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 **wFormatTag** value, located in the 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.

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

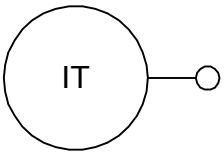

Figure 3-1: Input Terminal Icon

3.5.2 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 that endpoint and its associated Output Terminal. The class-specific endpoint descriptor contains a field that holds a direct reference to this Output Terminal. The Host needs to use both the 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 endpoint 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, 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 **wFormatTag** value, located in the 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.

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

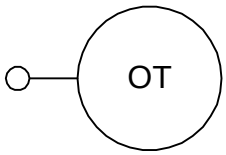

Figure 3-2: Output Terminal Icon

3.5.3 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 are n x m mixing 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 and by returning a Control setting of 0 dB when requested. Likewise, a missing connection can be implemented by reporting the Control as non-programmable and by returning a Control setting of -∞ dB.

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

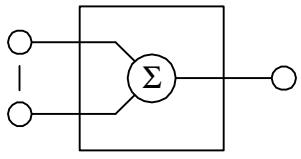

Figure 3-3: Mixer Unit Icon

3.5.4 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:

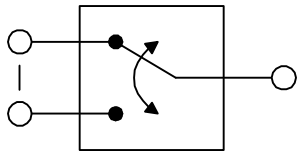

Figure 3-4: Selector Unit Icon

3.5.5 Feature Unit

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

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

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 which 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:

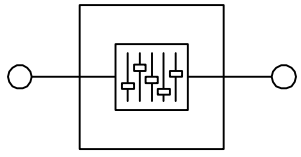

Figure 3-5: Feature Unit Icon

3.5.6 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 Processing Control, allowing the Host software to bypass whatever functionality is incorporated in the Processing Unit.

3.5.6.1 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 Processing 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:

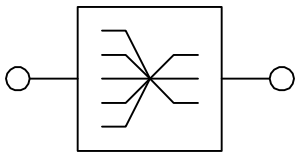

Figure 3-6: Up/Down-mix Processing Unit Icon

3.5.6.2 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 Processing 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:

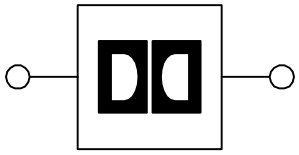

Figure 3-7: Dolby Prologic Processing Unit Icon

3.5.6.3 3D-Stereo Extender Processing Unit

The 3D-Stereo Extender Processing Unit operates on Left and Right channels only. It processes an existing stereo (two channel) soundtrack to add spaciousness and to make it appear to originate from outside the Left/Right speaker locations. Extended stereo effects can be achieved via various, straightforward methods. The algorithms and transforms applied to accomplish this are not defined by this specification and can be proprietary. The effects of the 3D-Stereo Extender Processing Unit can be bypassed at all times through manipulation of the Enable Processing Control. The size of the listening area (area in which the listener has to be placed with respect to speakers to hear the effect, also called sweet spot) can be controlled using the proper Get/Set Control request.

The symbol for the 3D-Stereo Extender Unit is depicted in the following figure:

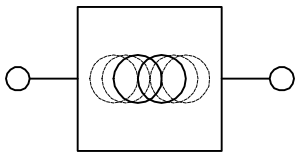

Figure 3-8: 3D-Stereo Extender Processing Unit Icon

3.5.6.4 Reverberation Processing Unit

The Reverberation Processing 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.
- Reverb Time: sets the time over which the reverberation will continue.
- Reverb Delay Feedback: used with Reverb Types Delay and Delay Panning. Sets the way in which delay repeats

The effects of the Reverberation Processing Unit can be bypassed at all times through manipulation of the Enable Processing Control.

In principle, the algorithm to produce the desired reverberation effect influences all channels as a whole. 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 Processing Unit can be found in the following figure:

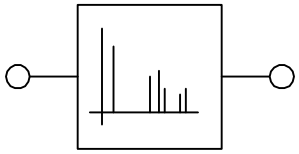

Figure 3-9: Reverberation Processing Unit Icon

3.5.6.5 Chorus Processing Unit

The Chorus Processing Unit is used to add chorus effects to the original audio information. A number of parameters can be manipulated to obtain the desired chorus effects.

- Chorus Level: controls the amount of the effect sound of chorus.
- Chorus Modulation Rate: sets the speed (frequency) of the modulator of the chorus.
- Chorus Modulation Depth: sets the depth at which the chorus sound is modulated.

The effects of the Chorus Processing Unit can be bypassed at all times through manipulation of the Enable Processing Control.

In principle, the algorithm to produce the desired chorus effect influences all channels as a whole. It is entirely left to the designer how a certain chorus effect is obtained. It is not the intention of this specification to precisely define all the parameters that influence the chorus experience.

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

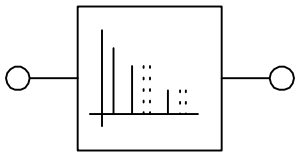

Figure 3-10: Chorus Processing Unit Icon

3.5.6.6 Dynamic Range Compressor Processing Unit

The Dynamic Range Compressor Processing 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.

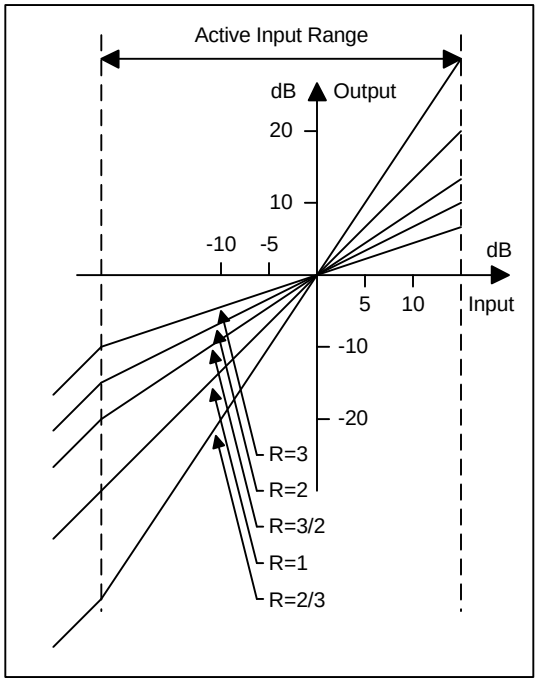

Figure 3-11: Dynamic Range Compressor Transfer Characteristic

- 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.

The effects of the Dynamic Range Compressor Processing Unit can be bypassed at all times through manipulation of the Enable Processing Control.

In principle, the algorithm to produce the desired dynamic range compression influences all channels as a whole. It is entirely left to the designer how a certain dynamic range compression is obtained.

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

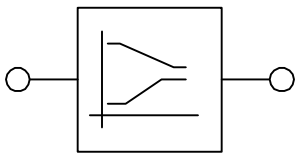

Figure 3-12: Dynamic Range Compressor Processing Unit Icon

3.5.7 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 Processing 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:

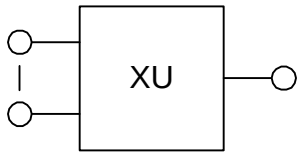

Figure 3-13: Extension Unit Icon

3.5.8 Associated Interfaces

In some cases, an audio function building block (Terminal, Mixer Unit, Feature Unit, and so on) needs to be associated with interfaces that are not part of the Audio Interface Collection. As an example, consider a speaker system with front-panel volume knob. The manufacturer might want to impose a binding between the front-panel volume Control and the speaker system's volume setting. The volume knob could be represented by a HID interface that coexists with the Audio Interface Collection. To create a binding between the Feature Unit inside the audio function that deals with master Volume Control and the front-panel volume knob, the Feature Unit descriptor can be supplemented by a special Associated Interface descriptor that holds a link to the associated HID interface.

In general, each Terminal or Unit descriptor can be supplemented by one or more optional Associated Interface descriptors that hold a reference to an interface. This interface is external to the audio function and interacts in a certain way with the Terminal or Unit. The layout of the Associated Interface descriptor is open-ended and is qualified by the Entity type it succeeds and by the target interface Class type it references.

For the time being, this specification does not define any specific Associated Interface descriptor layout.

3.6 Copy Protection

Because the Audio Device Class is primarily dealing with digital audio streams, the issue of protecting these – often-copyrighted – streams can not 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.

3.7 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 left and right channel samples. The receiving Input Terminal now splits the stream into a Left and Right logical channel. The AudioControl interface remains unchanged.

If the above AudioStreaming interface were an asynchronous sink, one extra isochronous synch endpoint would also be necessary.

Audio Interface Collections can be dynamic. Because the AudioControl interface, together with its associated AudioStreaming interface(s), constitute the 'logical interface' to the audio function, they must all come into existence at the same moment in time.

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.

3.7.1 AudioControl Interface

To control the functional behavior of a particular audio function, the Host can manipulate the 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 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 for status returns. This endpoint is optional.

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 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 may support multiple alternate settings. Alternate settings of the AudioControl interface could for instance be used to implement audio functions that support multiple topologies by presenting different class-specific AudioControl interface descriptors for each alternate setting.

3.7.1.1 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.

3.7.1.2 Status Interrupt Endpoint

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

The interrupt data is a 2-byte entity. The **bStatusType** field contains information in D7 indicating whether there is still an interrupt pending or not. This bit remains set until all pending interrupts are properly serviced. The other bits are used to report the cause of the interrupt in more detail. Bit D6 of the **bStatusType** field indicates a change in memory contents on one of the addressable Entities inside the audio function. This bit is cleared by a Get Memory request on the appropriate Entity. Bits D3..0 indicate the originator of the current interrupt. All addressable Entities inside an audio function can be originator.

The contents of the **bOriginator** field must be interpreted according to the code in D3..0 of the **bStatusType** field. If the originator is the AudioControl interface, the **bOriginator** field contains the TerminalID or UnitID of the Entity that caused the interrupt to occur. If the bOriginator field is set to zero, the 'virtual' Entity interface is the originator. This can be used to report global AudioControl interface changes to the Host. If the originator is an AudioStreaming interface, the **bOriginator** field contains the interface number of the AudioStreaming interface. Likewise, it contains the endpoint number if the originator were an AudioStreaming endpoint.

The proper response to an interrupt is either a Get Status request (D6=0) or a Get Memory request (D6=1). Issuing these requests to the appropriate originator must clear the Interrupt Pending bit and the Memory Contents Changed bit, if applicable.

The following table specifies the format of the status word:

Table 3-1: Status Word Format

Offset	Field	Size	Value	Description
0	bStatusType	1	Bitmap	D7: Interrupt Pending D6: Memory Contents Changed D5..4: Reserved D3..0: Originator 0 = AudioControl interface 1 = AudioStreaming interface 2 = AudioStreaming endpoint 3..15 = Reserved
1	bOriginator	1	Number	ID of the Terminal, Unit, interface, or endpoint that reports the interrupt.

3.7.2 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 synch endpoint for synchronization purposes. The isochronous data endpoint is required to be the first endpoint in the AudioStreaming interface. The synch endpoint always follows its associated data 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 bandwidth requirements an active AudioStreaming interface imposes on the USB. By incorporating a low-bandwidth or even zero-bandwidth alternate setting for each AudioStreaming interface, a device offers to the Host software the option to temporarily relinquish USB bandwidth by switching to this low-bandwidth alternate setting. If such an alternate setting is implemented, it must be the default alternate setting (alternate setting zero). A zero-bandwidth alternate setting can be implemented by specifying zero endpoints in the standard AudioStreaming interface descriptor. All other interface and endpoint descriptors (both standard and class-specific) need not be specified in this case.

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.

In addition, the concepts of dynamic interfaces as described in the *Universal Serial Bus Class Specification* can be used to notify the Host software that changes have occurred on the external connection. This is analogous to switching alternate settings on an AudioStreaming interface with USB endpoints, except that the switch is now device-initiated instead of Host-initiated.

As an example, consider an S/PDIF connection to an audio function. If nothing is connected to this external S/PDIF interface, the AudioStreaming interface is idle and reports itself as being dynamic and non-configured (bInterfaceClass=0x00). If the user connects a standard IEC958 signal to the audio function, the S/PDIF receiver inside the audio function detects this and notifies the Host that the AudioStreaming interface has switched to its IEC958 mode (alternate setting *x*). If, on the other hand, an

IEC1937 signal, carrying MPEG-encoded audio is connected, the AudioStreaming interface switches to the appropriate setting (alternate setting *y*) to handle the MPEG decoding process.

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.

3.7.2.1 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 a "stereo" audio data stream that contains audio data, encoded in Dolby Prologic format. Although there is only one data stream, carrying interleaved samples for Left and Right (or more precisely L_T and R_T), these two channels carry information for four logical channels (Left, Right, Center, and Surround). Other examples include cases in which multiple logical audio channels are compressed into a single data stream. The format of such a data stream can be entirely different from the native format of the logical channels (for example, 256 Kbits/s MPEG1 stereo audio as opposed to 176.4 Kbytes/s 16 bit stereo 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.

Consequently, 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 audio channel cluster by interleaving the individual audio data, and the result can be directed to the single endpoint. Furthermore, a single synch endpoint, if needed, can service the entire cluster. In this way, a minimum number of endpoints are consumed to transport related data streams.

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). If there is a need to manipulate a number of AudioStreaming interfaces as a whole, these interfaces can be tied together. The techniques for associating interfaces, described in the *Universal Serial Bus Class Specification* should be used to create the binding.

3.7.2.2 Isochronous Synch Endpoint

For adaptive audio source endpoints and asynchronous audio sink endpoints, an explicit synch mechanism is needed to maintain synchronization during transfers. For details about synchronization, see Section 5, "USB Data Flow Model," in the *USB Specification* and the relevant parts of the *Universal Serial Bus Class Specification*.

The information carried over the synch path consists of a 3-byte data packet. These three bytes contain the F_f value in a 10.14 format as described in Section 5.10.4.2, "Feedback" of the *USB Specification*. F_f represents the average number of samples the endpoint must produce or consume per frame to match the desired sampling frequency F_s exactly.

A new F_f value is available every $2^{(10-P)}$ ms (frames) where P can range from 1 to 9, inclusive. The sample clock F_s is always derived from a master clock F_m in the device. P is related to the ratio between those clocks through the following relationship:

$$F_m = F_s * 2^{(P-1)}$$

In worst case conditions, only F_s is available and $F_m = F_s$, giving $P = 1$ because one can always use phase information to resolve the estimation of F_s within half a clock cycle.

An adaptive audio source IN endpoint is accompanied by an associated isochronous synch OUT endpoint that carries F_f. An asynchronous audio sink OUT endpoint is accompanied by an associated isochronous synch IN endpoint.

For adaptive IN endpoints and asynchronous OUT endpoints, the standard endpoint descriptor provides the **bSynchAddress** field to establish a link to the associated synch endpoint. It contains the address of the synch endpoint. The **bSynchAddress** field of the synch standard endpoint descriptor must be set to zero.

As indicated earlier, a new F_f value is available every $2^{(10-P)}$ frames with P ranging from 1 to 9. The **bRefresh** field of the synch standard endpoint descriptor is used to report the exponent (10-P) to the Host. It can range from 9 down to 1. (512 ms down to 2 ms)

3.7.2.3 Audio Channel Cluster Format

An audio channel cluster is a grouping of logical audio channels that share the same characteristics like sampling frequency, bit resolution, etc. 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, effectively influencing all the channels at once. The maximum number of independent channels in an audio channel cluster is limited to 254. Indeed, Channel zero is used to reference the master channel and code 0xFF (255) is used in requests to indicate that the request parameter block holds values for all available addressed Controls. For further details, refer to Section 5.2.2, “AudioControl Requests” and the sections that follow, describing the second form of requests.

In many cases, each channel in the audio cluster is also tied to a certain location in the listening space. A trivial example of this is a cluster that contains Left and Right logical 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 logical channels in an audio channel cluster.

There are twelve predefined spatial locations:

- Left Front (L)
- Right Front (R)
- Center Front (C)
- Low Frequency Enhancement (LFE) [Super woofer]
- Left Surround (L_S)
- Right Surround (R_S)
- Left of Center (L_C) [in front]
- Right of Center (R_C) [in front]
- Surround (S) [rear]
- Side Left (S_L) [left wall]
- Side Right (S_R) [right wall]
- Top (T) [overhead]

If there are logical 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 logical channels Left, Right and LFE, then channel 1 is Left, channel 2 is Right, and channel 3 is LFE.

To characterize an audio channel cluster, a cluster descriptor is introduced. This descriptor is embedded within one of the following descriptors:

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

The cluster descriptor contains the following fields:

- **bNrChannels**: a number that specifies how many logical audio channels are present in the cluster.
- **wChannelConfig**: a bit field that indicates which spatial locations are present in the cluster. The bit allocations are as follows:

- D0: Left Front (L)
- D1: Right Front (R)
- D2: Center Front (C)
- D3: Low Frequency Enhancement (LFE)
- D4: Left Surround (L_S)
- D5: Right Surround (R_S)
- D6: Left of Center (L_C)
- D7: Right of Center (R_C)
- D8: Surround (S)
- D9: Side Left (S_L)
- D10: Side Right (S_R)
- D11: Top (T)
- D15..12: Reserved

- Each bit set in this bit map indicates there is a logical 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 **wChannelConfig** field, (i.e. **bNrChannels >** **[**Number_Of_Bits_Set]), then the first [Number_Of_Bits_Set] channels take the spatial positions, indicated in **wChannelConfig.** The remaining channels have ‘non-predefined’ spatial positions (positions that do not appear in the predefined list). If none of the bits in **wChannelConfig** 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)

Example 1:

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

Table 3-2: Dolby Prologic Cluster Descriptor

Offset	Field	Size	Value	Description

Offset	Field	Size	Value	Description
0	bNrChannels	1	4	There are 4 logical channels in the cluster.
1	wChannelConfig	2	0x0107	Left, Right, Center and Surround are present.
3	iChannelNames	1	Index	Because there are no non-predefined logical 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 3-3: Left Group Cluster Descriptor

Offset	Field	Size	Value	Description
0	bNrChannels	1	5	There are 5 logical channels in the cluster
1	wChannelConfig	2	0x0051	Left, Left Surround, Left of Center and two undefined channels are present. (bNrChannels > [Number_Of_Bits_Set])
3	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’
```

3.7.2.4 Audio Data Format

The format used to transport audio data over the USB is entirely determined by the code, located in the **wFormatTag** field of the class-specific interface descriptor. Therefore, each defined Format Tag must document in detail the audio data format it uses. Consequently, format-specific descriptors are needed to fully describe the format. For details about the predefined Format Tags 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.

4 Descriptors

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

4.1 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 audio-only devices, the device descriptor must indicate that class information is to be found at the interface level. Therefore, the **bDeviceClass** field of the device descriptor must contain zero so that enumeration software looks down at the interface level to determine the Interface Class. The **bDeviceSubClass** and **bDeviceProtocol** fields must be set to zero. All other fields of the device descriptor must comply with the definitions in Section 9.6.1, “Descriptor” of the *USB Specification*. There is no class-specific device descriptor.

4.2 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 9.6.2, “Configuration” of the *USB Specification*. There is no class-specific configuration descriptor.

4.3 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.

4.3.1 Standard AC Interface Descriptor

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

Table 4-1: 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.
4	bNumEndpoints	1	Number	Number of endpoints used by this interface (excluding endpoint 0). This number is either 0 or 1 if the optional status interrupt endpoint is present.

Offset	Field	Size	Value	Description
5	bInterfaceClass	1	Class	AUDIO. Audio Interface Class code (assigned by the USB). See Section A.1, “Audio Interface Class Code.”
6	bInterfaceSubClass	1	Subclass	AUDIOCONTROL. Audio Interface Subclass code. Assigned by this specification. See Section A.2, “Audio Interface Subclass Codes.”
7	bInterfaceProtocol	1	Protocol	Not used. Must be set to 0.
8	iInterface	1	Index	Index of a string descriptor that describes this interface.

4.3.2 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 Unit Descriptors (UDs) and Terminal Descriptors (TDs).

The total length of the class-specific AC interface descriptor depends on the number of 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 **bInCollection** field indicates how many AudioStreaming and MIDIStreaming interfaces there are in the Audio Interface Collection to which this AudioControl interface belongs. The **baInterfaceNr()** array contains the interface numbers of all the AudioStreaming and MIDIStreaming interfaces in the Collection. The **bInCollection** and **baInterfaceNr()** fields together provide all necessary information to determine which interfaces together constitute the entire USB interface to the audio function, i.e. describe the Audio Interface Collection.

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

Table 4-2: Class-Specific AC Interface Header Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	Number	Size of this descriptor, in bytes: 8+n
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.

Offset	Field	Size	Value	Description
5	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 Unit and Terminal descriptors.
7	bInCollection	1	Number	The number of AudioStreaming and MIDIStreaming interfaces in the Audio Interface Collection to which this AudioControl interface belongs: n
8	baInterfaceNr(1)	1	Number	Interface number of the first AudioStreaming or MIDIStreaming interface in the Collection.
…	…	…	…	…
8+(n-1)	baInterfaceNr(n)	1	Number	Interface number of the last AudioStreaming or MIDIStreaming interface in the Collection.

This header is followed by one or more Unit and/or Terminal Descriptors. The layout of the descriptors depends on the type of Unit or Terminal they represent. There are seven types of Unit and Terminal Descriptors possible. They are summarized in the following sections. The first four fields are common for all Unit and Terminal Descriptors. They contain the Descriptor Length, Descriptor Type, Descriptor Subtype, and Unit or Terminal ID.

Each Unit and Terminal within the audio function is assigned a unique identification number, the Unit ID (UID) or Terminal ID (TID), contained in the **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 (both 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.

4.3.2.1 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 same alternate setting of the AudioControl interface may have the same ID. This value must be passed in the **TerminalID** 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 the **bAssocTerminal** field is used, both associated Terminals must belong to the bi-directional Terminal Type group. 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 can not 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 **bNrChannels**, **wChannelConfig** 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 3.7.2.3, "Audio Channel Cluster Format."

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 4-3: Input 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	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	bNrChannels	1	Number	Number of logical output channels in the Terminal's output audio channel cluster.
8	wChannelConfig	2	Bitmap	Describes the spatial location of the logical channels.
10	iChannelNames	1	Index	Index of a string descriptor, describing the name of the first logical channel.
11	iTerminal	1	Index	Index of a string descriptor, describing the Input Terminal.

4.3.2.2 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 same alternate setting of the AudioControl interface may have the same ID. This value must be passed in the **UnitID** 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 the **bAssocTerminal** field is used, both associated Terminals must belong to the bi-directional Terminal Type group. 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 can not 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.

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 4-4: Output Terminal 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	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	iTerminal	1	Index	Index of a string descriptor, describing the Output Terminal.

4.3.2.3 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 same alternate setting of the AudioControl interface may have the same ID. This value must be passed in the **UnitID** 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.

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:

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

and *m* is the number of logical output channels, then there are *n* x *m* mixing Controls in the Mixer Unit, some of which may not be programmable.

Note: Both *n* and *m* must be limited to 254.

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**, **wChannelConfig** 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 3.7.2.3, "Audio Channel Cluster Format."

The Mixer Unit Descriptor reports which Controls are programmable in the **bmControls** 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, this means that the Mixer Unit contains a programmable mixing Control that connects input channel *u* to output channel *v*. If bit [*u*, *v*] is clear, this indicates that the connection between input channel *u* and output channel *v* is non-programmable. Its fixed value can be retrieved through the appropriate request. The valid range for *u* is from one to *n*. The valid range for *v* is from one to *m*.

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

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 4-5: Mixer Unit Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	Number	Size of this descriptor, in bytes: 10+p+N
1	bDescriptorType	1	Constant	CS_INTERFACE descriptor type.
2	bDescriptorSubtype	1	Constant	MIXER_UNIT descriptor subtype.

Offset	Field	Size	Value	Description
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	wChannelConfig	2	Bitmap	Describes the spatial location of the logical channels.
8+p	iChannelNames	1	Index	Index of a string descriptor, describing the name of the first logical channel.
9+p	bmControls	N	Number	Bit map indicating which mixing Controls are programmable.
9+p+N	iMixer	1	Index	Index of a string descriptor, describing the Mixer Unit.

4.3.2.4 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 same alternate setting of the AudioControl interface may have the same ID. This value must be passed in the **UnitID** 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 the same number of channels. However, the spatial locations of these channels may vary from cluster to cluster. Therefore, the Host software should trace all Input Pins to find their 'upstream' connection to locate the cluster descriptors for all the Input Pins that enter the Selector Unit. This further implies that the cluster descriptor, associated with the Output Pin of the Selector Unit can change dynamically, depending on the currently selected position of the Selector Unit.

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 4-6: Selector Unit Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	Number	Size of this descriptor, in bytes: 6+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	iSelector	1	Index	Index of a string descriptor, describing the Selector Unit.

4.3.2.5 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 same alternate setting of the AudioControl interface may have the same ID. This value must be passed in the **UnitID** 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.

The **bmaControls()** array is an array of bit-maps, each indicating the availability of certain audio Controls for a specific logical channel or for the master channel 0. For future expandability, the number of bytes occupied by each element (*n*) of the **bmaControls()** array is indicated in the **bControlSize** field. The number of logical channels in the cluster is denoted by *ch*.

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 4-7: Feature Unit Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	Number	Size of this descriptor, in bytes: 7+(ch+1)*n

Offset	Field	Size	Value	Description
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	bControlSize	1	Number	Size in bytes of an element of the bmaControls() array: n
6	bmaControls(0)	n	Bitmap	A bit set to 1 indicates that the mentioned Control is supported for master channel 0: D0: Mute D1: Volume D2: Bass D3: Mid D4: Treble D5: Graphic Equalizer D6: Automatic Gain D7: Delay D8: Bass Boost D9: Loudness D10..(n*8-1): Reserved
6+n	bmaControls(1)	n	Bitmap	A bit set to 1 indicates that the mentioned Control is supported for logical channel 1.
…	…	…	…	…
6+(ch*n)	bmaControls(ch)	n	Bitmap	A bit set to 1 indicates that the mentioned Control is supported for logical channel ch.
6+(ch+1)*n	iFeature	1	Index	Index of a string descriptor, describing this Feature Unit.

4.3.2.6 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 same alternate setting of the AudioControl interface may have the same ID. This value must be passed in the **UnitID** 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 Section A.7, “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**, **wChannelConfig,** 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 3.7.2.3, "Audio Channel Cluster Format."

The **bmControls** field is a bitmap, indicating the availability of certain audio Controls in the Processing Unit. For future expandability, the number of bytes occupied by the **bmControls** field is indicated in the **bControlSize** field. In general, all Controls are optional. However, some Processing Types may define certain Controls as mandatory. In such a case, the appropriate bit in the **bmControls** field must be set to one.

The meaning of the bits in the **bmControls** field is qualified by the **wProcessType** field. However, bit D0 always represents the Enable Processing Control for all Processing Unit Types. The Enable Processing Control is used to bypass the entire functionality of the Processing Unit. Default behavior is assumed when set to off. 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.

If the Enable Processing Control is present in a Processing Unit, bit D0 must be set to one. Otherwise, it is set to zero, indicating that the Processing Unit cannot be bypassed.

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 4-8: Common Part of the Processing Unit Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	Number	Size of this descriptor, in bytes: 13+p+n+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

Offset	Field	Size	Value	Description
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.
7+p+1	wChannelConfig	2	Bitmap	Describes the spatial location of the logical channels in the audio channel cluster of the Processing Unit.
7+p+3	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.
11+p	bControlSize	1	Number	Size in bytes of the bmControls field: n
12+p	bmControls	n	Bitmap	A bit set to 1 indicates that the mentioned Control is supported: D0: Enable Processing. D1..(n*8-1): process-specific allocation.
12+p +n	iProcessing	1	Index	Index of a string descriptor, describing this Processing Unit.
13+p +n	Process-specific	x	NA	A process-specific descriptor is appended to the common descriptor. See the following paragraphs.

4.3.2.6.1 Up/Down-mix Processing Unit Descriptor

The **wProcessType** field of the common Processing Unit descriptor contains the value UP/DOWNMIX_PROCESS. (See Appendix A.7, “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**, **wChannelConfig,** 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 is a bitmap, indicating the availability of certain audio Controls in the Up/Down-mix Processing Unit. For future expandability, the number of bytes occupied by the **bmControls** field is indicated in the **bControlSize** field.

Bit D0 of the **bmControls** field represents the Enable Processing Control. The Mode Select Control (D1) 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, **waModes()**. The index *i* into this array is one-based and directly related to the number of the mode described by entry **waModes(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 **waModes()** fields are very similar to those of the **wChannelConfig** field in a cluster descriptor (see Section 3.7.2.3, “Audio Channel Cluster Format”) i.e. a bit set in the **waModes(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 **waModes(i)** field can only contain set bits for those logical channels that are present in the output channel cluster. In other words, all **waModes()** fields can only contain a subset of the **wChannelConfig** field of the cluster descriptor of the Unit. Furthermore, logical channels that have a non-predefined spatial position can not be marked as active in the **waModes()** 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 4-9: Up/Down-mix Processing Unit Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	Number	Size of this descriptor, in bytes: 15+n+2*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	wChannelConfig	2	Bitmap	Describes the spatial location of the logical channels in the output channel cluster of the Processing Unit.
11	iChannelNames	1	Index	Index of a string descriptor, describing the name of the first logical channel in the Processing Unit’s output channel cluster.

Offset	Field	Size	Value	Description
12	bControlSize	1	Number	Size, in bytes, of the bmControls field: n
13	bmControls	n	Bitmap	A bit set to 1 indicates that the mentioned Control is supported: D0: Enable Processing. D1: Mode Select. D2..(n*8-1): Reserved
13+n	iProcessing	1	Index	Index of a string descriptor, describing this Processing Unit.
14+n	bNrModes	1	Number	Number of modes, supported by this Processing Unit: m
15+n	waModes(1)	2	Bitmap	Describes the active logical channels in mode 1.
…	…	…	…	…
15+n+(m-1)*2	waModes(m)	2	Bitmap	Describes active the logical channels in mode m.

4.3.2.6.2 Dolby Prologic Processing Unit Descriptor

The **wProcessType** field of the common Processing Unit descriptor contains the value DOLBY_PROLOGIC_PROCESS. (See Appendix A.7, “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**, **wChannelConfig**, 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.

Bit D0 of the **bmControls** field represents the Enable Processing Control. The Mode Select Control (D1) 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, **waModes()**. The bit allocations in the **waModes()** fields are very similar to those of the **wChannelConfig** field in a cluster descriptor (see Section 3.7.2.3, “Audio Channel Cluster Format”) i.e., a bit set in the **waModes(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 supports, at most, the following three different modes:

- Left, Right, Center channel decoding **waModes()** = 0x0007
- Left, Right, Surround channel decoding **waModes()** = 0x0103
- Left, Right, Center, Surround decoding **waModes()** = 0x0107

The **wChannelConfig** 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 4-10: Dolby Prologic Processing Unit Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	Number	Size of this descriptor, in bytes: 15+n+2*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.
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	wChannelConfig	2	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.
11	iChannelNames	1	Index	Index of a string descriptor, describing the name of the first logical channel in the Processing Unit's output channel cluster.
12	bControlSize	1	Number	Size, in bytes, of the bmControls field: n
13	bmControls	n	Bitmap	A bit set to 1 indicates that the mentioned Control is supported: D0: Enable Processing. D1: Mode Select. D2..(n*8-1): Reserved

Offset	Field	Size	Value	Description
13+n	iProcessing	1	Index	Index of a string descriptor, describing this Processing Unit.
14+n	bNrModes	1	Number	Number of modes, supported by this Processing Unit: m A maximum of 3 different modes is possible.
15+n	waModes(1)	2	Bitmap	Describes the active logical channels in mode 1.
…	…	…	…	…
15+n+(m-1)*2	waModes(m)	2	Bitmap	Describes active the logical channels in mode m.

4.3.2.6.3 3D-Stereo Extender Processing Unit Descriptor

The **wProcessType** field of the common Processing Unit descriptor contains the value 3D-STEREO_EXTENDER_PROCESS. (See Section A.7, “Processing Unit Process Types”)

The 3D-Stereo Extender Processing Unit has a single Input Pin. Therefore, the **bNrInputs** field must contain the value 1. The **bNrChannels**, **wChannelConfig** and **iChannelNames** fields together constitute the output cluster descriptor of the 3D-Stereo Extender Processing Unit. It describes which logical channels are physically present at the output of the Processing Unit.

Bit D0 of the **bmControls** field represents the Enable Processing Control. Bit D1 indicates the availability of the Spaciousness Control.

Although the input cluster may contain a number of logical channels, the 3D-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 discarded 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 3D-Stereo Extender Processing Unit.

The following table outlines the 3D-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 4-11: 3D-Stereo Extender Processing Unit Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	Number	Size of this descriptor, in bytes: 14+n
1	bDescriptorType	1	Constant	CS_INTERFACE descriptor type.
2	bDescriptorSubtype	1	Constant	PROCESSING_UNIT descriptor subtype.

Offset	Field	Size	Value	Description
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	3D-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	wChannelConfig	2	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.
11	iChannelNames	1	Index	Index of a string descriptor, describing the name of the first logical channel in the Processing Unit's output channel cluster.
12	bControlSize	1	Number	Size, in bytes, of the bmControls field: n
13	bmControls	n	Bitmap	A bit set to 1 indicates that the mentioned Control is supported: D0: Enable Processing. D1: Spaciousness. D2..(n*8-1): Reserved
13+n	iProcessing	1	Index	Index of a string descriptor, describing this Processing Unit.

4.3.2.6.4 Reverberation Processing Unit Descriptor

The **wProcessType** field of the common Processing Unit descriptor contains the value REVERBERATION_PROCESS. (See Section A.7, "Processing Unit Process Types")

The Reverberation Processing Unit has a single Input Pin. Therefore, the **bNrInputs** field must contain the value 1. The **bNrChannels**, **wChannelConfig** and **iChannelNames** fields together constitute the output cluster descriptor of the Reverberation Processing Unit. It describes which logical channels are physically present at the output of the Processing Unit. In most cases, this will be identical to the configuration of the input channel cluster.

The **bmControls** field indicates which reverberation-related Controls are effectively implemented in the Reverberation Processing Unit.

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

Table 4-12: Reverberation Processing Unit Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	Number	Size of this descriptor, in bytes: 14+n
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	REVERBERATION_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	wChannelConfig	2	Bitmap	Describes the spatial location of the logical channels in the output channel cluster of the Processing Unit.
11	iChannelNames	1	Index	Index of a string descriptor, describing the name of the first logical channel in the output channel cluster Processing Unit.
12	bControlSize	1	Number	Size, in bytes, of the bmControls field: n
13	bmControls	n	Bitmap	A bit set to 1 indicates that the mentioned Control is supported: D0: Enable Processing. D1: Reverb Type. D2: Reverb Level. D3: Reverb Time. D4: Reverb Delay Feedback. D5..(n*8-1): Reserved.
13+n	iProcessing	1	Index	Index of a string descriptor, describing this Processing Unit.

4.3.2.6.5 Chorus Processing Unit Descriptor

The **wProcessType** field of the common Processing Unit descriptor contains the value CHORUS_PROCESS. (See Section A.7, “Processing Unit Process Types”)

The Chorus Processing Unit has a single Input Pin. Therefore, the **bNrInputs** field must contain the value 1. The **bNrChannels**, **wChannelConfig**, and **iChannelNames** fields together constitute the output cluster

descriptor of the Chorus Processing Unit. It describes which logical channels are physically present at the output of the Processing Unit. In most cases, this will be identical to the configuration of the input channel cluster.

The **bmControls** field indicates which chorus-related Controls are effectively implemented in the Chorus Processing Unit.

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

Table 4-13: Chorus Processing Unit Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	Number	Size of this descriptor, in bytes: 14+n
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	CHORUS_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	wChannelConfig	2	Bitmap	Describes the spatial location of the logical channels in the output channel cluster of the Processing Unit.
11	iChannelNames	1	Index	Index of a string descriptor, describing the name of the first logical channel in the output channel cluster of the Processing Unit.
12	bControlSize	1	Number	Size, in bytes, of the bmControls field: n
13	bmControls	n	Bitmap	A bit set to 1 indicates that the mentioned Control is supported: D0: Enable Processing. D1: Chorus Level. D2: Chorus Modulation Rate. D3: Chorus Modulation Depth. D4..(n*8-1): Reserved

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

4.3.2.6.6 Dynamic Range Compressor Processing Unit Descriptor

The **wProcessType** field of the common Processing Unit descriptor contains the value DYN_RANGE_COMP_PROCESS. (See Section A.7, “Processing Unit Process Types”)

The Dynamic Range Compressor Processing Unit has a single Input Pin. Therefore, the **bNrInputs** field must contain the value 1. The **bNrChannels**, **wChannelConfig**, and **iChannelNames** fields together constitute the output cluster descriptor of the Dynamic Range Compressor Processing Unit. It describes which logical channels are physically present at the output of the Processing Unit. In most cases, this will be identical to the configuration of the input channel cluster.

The **bmControls** field indicates which Controls are effectively implemented in the Dynamic Range Compressor Processing Unit.

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

Table 4-14: Dynamic Range Compressor Processing Unit Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	Number	Size of this descriptor, in bytes: 14+n
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	DYN_RANGE_COMP_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	wChannelConfig	2	Bitmap	Describes the spatial location of the logical channels in the output channel cluster of the Processing Unit.

Offset	Field	Size	Value	Description
11	iChannelNames	1	Index	Index of a string descriptor, describing the name of the first logical channel in the output channel cluster of the Processing Unit.
12	bControlSize	1	Number	Size, in bytes, of the bmControls field: n
13	bmControls	n	Bitmap	A bit set to 1 indicates that the mentioned Control is supported: D0: Enable Processing. D1: Compression Ratio. D2: MaxAmpl. D3: Threshold. D4: Attack time. D5: Release time. D6..(n*8-1): Reserved
13+n	iProcessing	1	Index	Index of a string descriptor, describing this Processing Unit.

4.3.2.7 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 same alternate setting of the AudioControl interface may have the same ID. This value must be passed in the **UnitID** 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**, **wChannelConfig**, 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 3.7.2.3, "Audio Channel Cluster Format."

The **bmControls** field is a bitmap, indicating the availability of certain audio Controls in the Extension Unit. For future expandability, the number of bytes occupied by the **bmControls** field is indicated in the **bControlSize** field. In general, all Controls are optional, except for the Enable Processing Control. This Control must be supported by every Extension Unit.

The Enable Processing 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 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 by the Extension 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 Extension Unit.

The following table outlines the Extension Unit descriptor.

Table 4-15: Extension 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	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.
…	…	…	…	…
7+(p-1)	baSourceID (p)	1	Number	ID of the Unit or Terminal to which the last Input Pin of this Extension Unit is connected.
7+p	bNrChannels	1	Number	Number of logical output channels in the audio channel cluster of the Extension Unit.
7+p+1	wChannelConfig	2	Bitmap	Describes the spatial location of the logical channels in the audio channel cluster of the Extension Unit.
7+p+3	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.
11+p	bControlSize	1	Number	Size, in bytes, of the bmControls field: n

Offset	Field	Size	Value	Description
12+p	bmControls	n	Bitmap	A bit set to 1 indicates that the mentioned Control is supported: D0 Enable Processing D1..(n*8-1) Reserved
12+p+n	iExtension	1	Index	Index of a string descriptor, describing this Extension Unit.

4.3.2.8 Associated Interface Descriptor

The Associated Interface descriptor provides a means to indicate a relationship between a Terminal or a Unit and an interface, external to the audio function. It directly follows the Entity descriptor to which it is related.

The **bInterfaceNr** field contains the interface number of the associated interface. The remainder of the descriptor depends both on the Entity to which it is related and on the interface class of the target interface. At this moment, no specific layouts are defined by this specification.

The following table outlines the Associated Interface descriptor.

Table 4-16: Associated Interfaces Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	Number	Size of this descriptor, in bytes: 4+x
1	bDescriptorType	1	Constant	CS_INTERFACE descriptor type.
2	bDescriptorSubtype	1	Constant	ASSOC_INTERFACE descriptor subtype.
3	bInterfaceNr	1	Number	The interface number of the associated interface.
4	Association-specific	x	Number	Association-specific extension to the open-ended descriptor.

4.4 AudioControl Endpoint Descriptors

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

4.4.1 AC Control Endpoint Descriptors

4.4.1.1 Standard AC Control Endpoint Descriptor

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

4.4.1.2 Class-Specific AC Control Endpoint Descriptor

There is no dedicated class-specific control endpoint descriptor.

4.4.2 AC Interrupt Endpoint Descriptors

4.4.2.1 Standard AC Interrupt Endpoint Descriptor

The interrupt endpoint descriptor is identical to the standard endpoint descriptor defined in Section 9.6.4, "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 4-17: Standard AC Interrupt Endpoint Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	Number	Size of this descriptor, in bytes: 9
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	D3..2: Synchronization type 00 = None 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 2-byte status information. Set to 2 if not shared, set to the appropriate value if shared.
6	bInterval	1	Number	Left to the designer's discretion. A value of 10 ms or more seems sufficient.
7	bRefresh	1	Number	Reset to 0.
8	bSynchAddress	1	Endpoint	Reset to 0.

4.4.2.2 Class-Specific AC Interrupt Endpoint Descriptor

There is no class-specific AudioControl interrupt endpoint descriptor.

4.5 AudioStreaming Interface Descriptors

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

4.5.1 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 4-18: 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).
5	bInterfaceClass	1	Class	AUDIO Audio Interface Class code (assigned by the USB). See Section A.1, “Audio Interface Class Code.”
6	bInterfaceSubClass	1	Subclass	AUDIO_STREAMING Audio Interface Subclass code. Assigned by this specification. See Section A.2, “Audio Interface Subclass Codes.”
7	bInterfaceProtocol	1	Protocol	Not used. Must be set to 0.
8	iInterface	1	Index	Index of a string descriptor that describes this interface.

4.5.2 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 **bDelay** field holds a value that is a measure for the delay that is introduced in the audio data stream due to internal processing of the signal within the audio function. The Host software can take this value into account when phase relations between audio streams, processed by different audio functions, are important.

The **wFormatTag** field holds information about the Audio Data Format that should be used when communicating with this interface. If the interface has a USB isochronous endpoint associated with it, the **wFormatTag** field describes the Audio Data Format that should be used when exchanging data with this endpoint. If the interface has no endpoint, the **wFormatTag** field describes the Audio Data Format that is used on the (external) connection this interface represents.

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 Audio Data Formats is provided in a separate document, *USB Audio Data Formats*, that is considered part of this specification. Further specific

information concerning the Audio Data Format for this interface is reported in a separate type-specific descriptor, see Section 4.5.3, “Class-Specific AS Format Type Descriptor.” This can optionally be supplemented by format-specific information through a format-specific descriptor, see Section 4.5.4, “Class-Specific AS Format-Specific Descriptor.”

Table 4-19: Class-Specific AS Interface 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	AS_GENERAL descriptor subtype.
3	bTerminalLink	1	Constant	The Terminal ID of the Terminal to which the endpoint of this interface is connected.
4	bDelay	1	Number	Delay (δ) introduced by the data path (see Section 3.4, “Inter Channel Synchronization”). Expressed in number of frames.
5	wFormatTag	2	Number	The Audio Data Format that has to be used to communicate with this interface.

4.5.3 Class-Specific AS Format Type Descriptor

The **wFormatTag** field in the class-specific AS Interface Descriptor implicitly indicates which Format Type should be used to communicate with the connection (USB or external) this interface represents. (Each Audio Data Format belongs to a certain Format Type as outlined in *USB Audio Data Formats*.)

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 *USB Audio Data Formats*.

4.5.4 Class-Specific AS Format-Specific Descriptor

As stated earlier, the **wFormatTag** field in the class-specific AS Interface Descriptor not only describes to what Format Type the interface belongs. It also states exactly what Audio Data Format should be used to communicate with the connection (USB or external) this interface represents. Some Audio Data Formats need additional format-specific information conveyed to the Host. Therefore, the Format Type descriptor may be followed by a class-specific AS format-specific descriptor. The details and layout of this descriptor for the Audio Data Formats that need it, is outlined in *USB Audio Data Formats*.

4.6 AudioStreaming Endpoint Descriptors

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

4.6.1 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.

4.6.1.1 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 9.6.4, "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 = 1) or an audio sink (D7 = 0). 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 5.10.4.1, "Synchronous Type," of the *USB Specification*.

Table 4-20: Standard AS Isochronous Audio Data Endpoint Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	Number	Size of this descriptor, in bytes: 9
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. 0 = OUT endpoint 1 = IN endpoint D6..4: Reserved, reset to zero D3..0: The endpoint number, determined by the designer.
3	bmAttributes	1	Bit Map	D3..2: Synchronization type 01 = Asynchronous 10 = Adaptive 11 = Synchronous D1..0: Transfer type 01 = Isochronous 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 expressed in milliseconds. Must be set to 1.
7	bRefresh	1	Number	Reset to 0.

Offset	Field	Size	Value	Description
8	bSynchAddress	1	Endpoint	The address of the endpoint used to communicate synchronization information if required by this endpoint. Reset to zero if no synchronization pipe is used.

4.6.1.2 Class-Specific AS Isochronous Audio Data Endpoint Descriptor

The **bmAttributes** field indicates which endpoint-specific Controls this endpoint supports through bits D6..0. Bit D7 is reserved to indicate whether the endpoint always needs USB packets of **wMaxPacketSize** length (D7 = 1) or that it can handle short packets (D7 = 0). 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 more bytes than expected and discard the superfluous zero bytes.

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 lead to either fixed time or fixed number of samples lock delay. 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 4-21: Class-Specific 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	CS_ENDPOINT descriptor type.
2	bDescriptorSubtype	1	Constant	EP_GENERAL descriptor subtype.
3	bmAttributes	1	Bit Map	A bit in the range D6..0 set to 1 indicates that the mentioned Control is supported by this endpoint. D0: Sampling Frequency D1: Pitch D6..2: Reserved Bit D7 indicates a requirement for **wMaxPacketSize** packets. D7: MaxPacketsOnly

Offset	Field	Size	Value	Description
4	bLockDelayUnits	1	Number	Indicates the units used for the wLockDelay field: 0: Undefined 1: Milliseconds 2: Decoded PCM samples 3..255: Reserved
5	wLockDelay		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.

4.6.2 AS Isochronous Synch 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.

4.6.2.1 Standard AS Isochronous Synch Endpoint Descriptor

The isochronous synch endpoint descriptor is identical to the standard endpoint descriptor defined in Section 9.6.4, “Endpoint,” of the *USB Specification* and further expanded as defined in the *Universal Serial Bus Class Specification*. The **bmAttributes** field bits are set to reflect the isochronous type and synchronization type of the endpoint.

Table 4-22: Standard AS Isochronous Synch Endpoint Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	Number	Size of this descriptor, in bytes: 9
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. 0 = OUT endpoint for sources 1 = IN endpoint for sinks D6..4: Reserved, reset to zero D3..0: The endpoint number, determined by the designer.
3	bmAttributes	1	Bit Map	D3..2: Synchronization type 00 = None D1..0: Transfer type 01 = Isochronous 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.

Offset	Field	Size	Value	Description
6	bInterval	1	Number	Interval for polling endpoint for data transfers expressed in milliseconds. Must be set to 1.
7	bRefresh	1	Number	This field indicates the rate at which an isochronous synchronization pipe provides new synchronization feedback data. This rate must be a power of 2, therefore only the power is reported back and the range of this field is from 1 (2 ms) to 9 (512 ms).
8	bSynchAddress	1	Endpoint	Must be reset to zero.

4.6.2.2 Class-Specific AS Isochronous Synch Endpoint Descriptor

There is no class-specific AS isochronous synch 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

Most 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 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 Units of the audio function. The class-specific AudioControl interface descriptor contains a collection of Unit descriptors, each indicating which Controls are present in every Unit. AudioControl requests are always directed to the single AudioControl interface of the audio function. The request contains enough information (Unit ID, Channel Number, and Control Selector) 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 standard Controls, 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 additional class-specific request:

- **Memory Requests**. Every addressable Entity in the audio function (Terminal, Unit, 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.
- The **Get Status** request is a general query to an Entity in the AudioControl interface or one of the Audio Streaming interfaces and does not manipulate Controls.

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.

The rest of this section describes the class-specific requests used to manipulate both audio Controls and endpoint Controls.

5.2.1 Request Layout

The following paragraphs describe the general structure of the Set and Get requests. Subsequent paragraphs detail the use of the Set/Get requests for the different request types.

5.2.1.1 Set Request

This request is used to set an attribute of a Control inside an Entity of the audio function. Additionally, the memory space attribute of an Entity itself can be set through this request.

Table 5-1: Set Request Values

bmRequest Type	bRequest	wValue	wIndex	wLength	Data
00100001B	SET_CUR SET_MIN SET_MAX SET_RES SET_MEM	See following paragraphs	Entity ID and Interface	Length of parameter block	Parameter block
00100010B			Endpoint		

The **bmRequestType** field specifies that this is a SET request (D7=0b0). It is a class-specific request (D6..5=0b01), directed to either the AudioControl interface or an AudioStreaming interface 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 or Entity is to be modified. Possible attributes for a Control are its:

- Current setting attribute (SET_CUR)
- Minimum setting attribute (SET_MIN)
- Maximum setting attribute (SET_MAX)
- Resolution attribute (SET_RES)

Possible attributes for an Entity are its:

- Memory space attribute (SET_MEM)

If the addressed Control or Entity 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 and MEM attribute will be supported for the Set request. However, this specification does not prevent a designer from making other attributes programmable. For the list of Request constants, refer to Section A.9, “Audio Class-Specific Request Codes.”

The **wValue** field interpretation is qualified by the value in the **wIndex** field. Depending on what Entity is addressed, the layout of the **wValue** field changes. The following paragraphs describe the contents of the **wValue** field for each Entity separately. In most cases, the **wValue** field contains the Control Selector (CS) in the high byte. It is used to address a particular Control within Entities that can contain multiple Controls. If the Entity only contains a single Control, there is no need to specify a Control Selector and the **wValue** field can be used to pass additional parameters.

The **wIndex** field specifies the interface or endpoint to be addressed in the low byte and the Entity 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 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.

5.2.1.2 Get Request

This request returns the attribute setting of a specific Control inside an Entity of the audio function. Additionally, the memory space attribute of an Entity itself can be returned through this request.

Table 5-2: Get Request Values

bmRequest Type	bRequest	wValue	wIndex	wLength	Data
10100001B	GET_CUR GET_MIN GET_MAX GET_RES GET_MEM	See following paragraphs	Entity ID and Interface	Length of parameter block	Parameter block
10100010B			Endpoint		

The **bmRequestType** field specifies that this is a GET request (D7=0b1). It is a class-specific request (D6..5=0b01), directed to either the AudioControl interface or an AudioStreaming interface 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 or Entity is to be returned. Possible attributes for a Control are its:

- Current setting attribute (GET_CUR)
- Minimum setting attribute (GET_MIN)
- Maximum setting attribute (GET_MAX)
- Resolution attribute (GET_RES)

Possible attributes for an Entity are its:

- Memory space attribute (GET_MEM)

If the addressed Control or Entity does not support readout of a certain attribute, the control pipe must indicate a stall when an attempt is made to read that attribute. For the list of Request constants, refer to Section A.9, “Audio Class-Specific Request Codes.”

The **wValue** field interpretation is qualified by the value in the **wIndex** field. Depending on what Entity is addressed, the layout of the **wValue** field changes. The following paragraphs describe the contents of the **wValue** field for each Entity separately. In most cases, the **wValue** field contains the Control Selector (CS) in the high byte. It is used to address a particular Control within Entities that can contain multiple Controls. If the Entity only contains a single Control, there is no need to specify a Control Selector and the **wValue** field can be used to pass additional parameters.

The **wIndex** field specifies the interface or endpoint to be addressed in the low byte and the Entity 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 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.

5.2.2 AudioControl Requests

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

5.2.2.1 Terminal Control Requests

The following paragraphs describe the Set and Get Terminal Control requests.

5.2.2.1.1 Set Terminal Control Request

This request is used to set an attribute of a Terminal Control inside a Terminal of the audio function.

Table 5-3: Set Terminal Control Request Values

bmRequest Type	bRequest	wValue	wIndex	wLength	Data
00100001B	SET_CUR SET_MIN SET_MAX SET_RES	CS	Terminal ID and Interface	Length of parameter block	Parameter block

The **bRequest** field indicates which attribute the request is manipulating. The MIN, MAX and RES attributes are usually not supported for the Set request.

The **wValue** field specifies the Control Selector (CS) in the high byte and zero in the low byte. The Control Selector indicates which type of Control this request is manipulating. If the request specifies an unknown or unsupported CS to that Terminal, the control pipe must indicate a stall.

For a description of the parameter block for the Terminal Controls, see Section 5.2.2.1.3, “Terminal Controls.”

5.2.2.1.2 Get Terminal Control Request

This request returns the attribute setting of a specific Terminal Control inside a Terminal of the audio function.

Table 5-4: Get Terminal Control Request Values

bmRequest Type	bRequest	wValue	wIndex	wLength	Data
10100001B	GET_CUR GET_MIN GET_MAX GET_RES	CS	Terminal ID and Interface	Length of parameter block	Parameter block

The **bRequest** field indicates which attribute the request is reading

The **wValue** field specifies the Control Selector (CS) in the high byte and zero in the low byte. The Control Selector indicates which type of Control this request is addressing. If the request specifies an unknown or unsupported CS to that Terminal, the control pipe must indicate a stall.

For a description of the parameter block for the Terminal Controls, see Section 5.2.2.1.3, “Terminal Controls.”

5.2.2.1.3 Terminal Controls

For the time being, this specification defines only one Terminal Control. The following section presents a detailed description.

5.2.2.1.3.1 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 only supports 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 Copy Protect Control honors the request to the best of its abilities. It may round the **bCopyProtect** attribute value to its closest available setting. It will report this setting when queried during a Get Control request.

Table 5-5: Copy Protect Control Parameter Block

Control Selector		COPY_PROTECT_CONTROL		
wLength		1		
Offset	**Field**	**Size**	**Value**	**Description**
0	bCopyProtect	1	Number	The setting for the CUR attribute of the CopyProtect Control. 0x00: CPL0 0x01: CPL1 0x02: CPL2

5.2.2.2 Mixer Unit Control Requests

The following paragraphs describe the Set and Get Mixer Unit Control requests. The Set Mixer Unit Control request can have two forms. The first form must be supported while the second form can be optionally implemented. The Get Mixer Unit Control request can have three forms. The first form must be supported, while the second and third form can be optionally implemented. If the second form of the Set request is implemented, the second form of the Get request must also be implemented.

5.2.2.2.1 Set Mixer Unit Control Request

This request is used to set an attribute of a Mixer Control inside a Mixer Unit of the audio function.

Table 5-6: Set Mixer Unit Control Request Values

bmRequest Type	bRequest	wValue	wIndex	wLength	Data
00100001B	SET_CUR SET_MIN SET_MAX SET_RES	ICN and OCN	Mixer Unit ID and Interface	Length of parameter block	Parameter block

The **bRequest** field indicates which attribute the request is manipulating. The MIN, MAX and RES attributes are usually not supported for the Set request.

Because the Mixer Unit only contains a single type of Control, there is no need for a Control Selector. Instead, the **wValue** field specifies the Input Channel Number (ICN) in the high byte and the Output Channel Number (OCN) in the low byte of the Mixer Control to be influenced. The ICN is derived from both the input cluster number and the logical channel number within that cluster, according to the rules established in Section 4.3.2.3, “Mixer Unit Descriptor.” (ICN=*u*, OCN=*v*) If the request specifies an unknown ICN or OCN to that Unit or refers to a non-programmable Mixer Control in the Mixer Unit, the control pipe must indicate a stall.

A special case arises when the Input Channel Number and Output Channel Number are both set to 0xFF. Then a single Set Mixer Unit Control request can be used to set an attribute of all the programmable Mixer Controls within the Unit. The number of parameters passed in the parameter block must exactly match the number of programmable Mixer Controls in the Unit. If this is not the case, the control pipe must indicate a stall. The ordering of the parameters in the parameter block obeys the same rules as established for the bit ordering in the **bmControls** field of the Mixer Unit Descriptor. The parameter block must contain an attribute setting for every Mixer Control that has its bit set in the **bmControls** field. The previous description is referred to as the second form of the Set Mixer Unit Control request.

For a description of the parameter block for the Set Mixer Unit Control request, see Section 5.2.2.2.3, “Mixer Control.”

5.2.2.2.2 Get Mixer Unit Control Request

This request returns the attribute setting of a specific Mixer Control inside a Mixer Unit of the audio function.

Table 5-7: Get Mixer Unit Control Request Values

bmRequest Type	bRequest	wValue	wIndex	wLength	Data
10100001B	GET_CUR GET_MIN GET_MAX GET_RES	ICN and OCN	Mixer Unit ID and Interface	Length of parameter block	Parameter block

The **bRequest** field indicates which attribute the request is reading

Because the Mixer Unit only contains a single type of Control, there is no need for a Control Selector. Instead, the **wValue** field specifies the Input Channel Number (ICN) in the high byte and the Output Channel Number (OCN) in the low byte of the Mixer Control to be retrieved. The ICN is derived from both the input cluster number and the logical channel number within that cluster, according to the rules

established in Section 4.3.2.3, “Mixer Unit Descriptor.” (ICN=*u*, OCN=*v*) If the request specifies an unknown ICN or OCN to that Unit, the control pipe must indicate a stall. For the Get Mixer Unit request, it is legal to address a non-programmable Mixer Control.

A special case arises when the Input Channel Number and Output Channel Number are both set to 0xFF. Then a single Get Mixer Unit Control request can be used to retrieve an attribute setting of all the programmable Mixer Controls within the Unit. The ordering of the parameters in the parameter block obeys the same rules as established for the bit ordering in the **bmControls** field of the Mixer Unit Descriptor. The previous description is referred to as the second form of the Get Mixer Unit Control request.

Another special case arises when the Input Channel Number and Output Channel Number are both set to 0x00. Then a single Get Mixer Unit Control request can be used to retrieve an attribute setting of all the Mixer Controls (both programmable and non-programmable) within the Unit. The ordering of the parameters in the parameter block obeys the same rules as established for the bit ordering in the **bmControls** field of the Mixer Unit Descriptor. The parameter block now contains a setting for every Mixer in the Mixer Unit. The above description is referred to as the third form of the Get Mixer Unit Control request.

For a description of the parameter block for the Get Mixer Unit Control request, see Section 5.2.2.2.3, “Mixer Control.”

5.2.2.2.3 Mixer Control

A Mixer Unit consists of a number of Mixer Controls, either programmable or fixed. A Mixer Control can support all possible Control attributes (CUR, MIN, MAX, and RES). 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 range for the CUR attribute is extended by code 0x8000, representing silence, i.e., -∞ dB. The settings for the RES attribute can only take positive values and range from 1/256 dB (0x0001) to +127.9961 dB (0x7FFF). The Mixer Control honors the request to the best of its abilities. It may round the **wMixer** attribute value to its closest available setting. It will report this rounded setting when queried during a Get Mixer Unit Control request.

In the first form of the Mixer Unit Control request, a particular Mixer Control within a Mixer Unit is addressed through the Unit ID, Input Channel Number, and Output Channel Number fields of the Set/Get Mixer Unit Control request.

Table 5-8: First Form of the Mixer Control Parameter Block

wLength	2

Offset	Field	Size	Value	Description
0	wMixer	2	Number	The setting for the attribute of the addressed Mixer Control: 0x7FFF: 127.9961 dB … 0x0100: 1.0000 dB … 0x0002: 0.0078 dB 0x0001: 0.0039 dB 0x0000: 0.0000 dB 0xFFFF: -0.0039 dB 0xFFFE: -0.0078 dB … 0xFE00: -1.0000 dB … 0x8002: -127.9922 dB 0x8001: -127.9961 dB 0x8000 -∞ dB (CUR attribute only)

In the second form, the Input and Output Channel Number fields are both set to 0xFF. The parameter block contains a list of settings for an attribute of all programmable Mixer Controls in the Mixer Unit.

Table 5-9: Second Form of the Mixer Control Parameter Block

wLength		(Number of programmable Controls: NrPr)*2		
Offset	**Field**	**Size**	**Value**	**Description**
0	wMixer(1)	2	Number	The setting for the attribute of the first programmable Mixer Control.
…	…	…	…	…
(NrPr-1)*2	wMixer(NrPr)	2	Number	The setting for the attribute of the last programmable Mixer Control.

In the third form, the Input and Output Channel Number fields are both set to 0x00. The parameter block contains a list of settings for an attribute of all the Mixer Controls in the Mixer Unit.

Table 5-10: Third Form of the Mixer Control Parameter Block

wLength		(Number of Controls: NrCo)*2		
Offset	**Field**	**Size**	**Value**	**Description**
0	wMixer(1)	2	Number	The setting for the attribute of the first Mixer Control.
…	…	…	…	…
(NrCo-1)*2	wMixer(NrCo)	2	Number	The mixer setting for the attribute of the last Mixer Control.

5.2.2.3 Selector Unit Control Requests

The following paragraphs describe the Set and Get Selector Unit Control requests.

5.2.2.3.1 Set Selector Unit Control Request

This request is used to set an attribute of a Selector Control inside a Selector Unit of the audio function.

Table 5-11: Set Selector Unit Control Request Values

bmRequest Type	bRequest	wValue	wIndex	wLength	Data
00100001B	SET_CUR SET_MIN SET_MAX SET_RES	Zero	Selector Unit ID and Interface	Length of parameter block	Parameter block

The **bRequest** field indicates which attribute the request is manipulating. The MIN, MAX, and RES attributes are usually not supported for the Set request.

The **wValue** field is not used and must be set to zero (Because the Selector Unit only contains a single Control, there is no need for a Control Selector).

For a description of the parameter block for the Set Selector Unit Control request, see Section 5.2.2.3.3, “Selector Control.”

5.2.2.3.2 Get Selector Unit Control Request

This request returns the attribute setting of the Selector Control inside a Selector Unit of the audio function.

Table 5-12: Get Selector Unit Control Request Values

bmRequest Type	bRequest	wValue	wIndex	wLength	Data
10100001B	GET_CUR GET_MIN GET_MAX GET_RES	Zero	Selector Unit ID and Interface	Length of parameter block	Parameter block

The **bRequest** field indicates which attribute the request is reading. The RES attribute is usually not supported and should return one if implemented.

The **wValue** field is not used and must be set to zero.

For a description of the parameter block for the Get Selector Unit Control request, see Section 5.2.2.3.3, “Selector Control.”

5.2.2.3.3 Selector Control

A Selector Unit represents a multi-channel source selector, capable of selecting between a number of identically configured audio channel clusters. The valid range for the CUR, MIN, and MAX attributes 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 RES attribute can only have a value of one. The Selector Control honors the request to the best of its abilities. It may round the **bSelector** attribute value to its closest

available setting. It will report this rounded setting when queried during a Get Selector Unit Control request.

Table 5-13: Selector Control Parameter Block

wLength		1		
Offset	**Field**	**Size**	**Value**	**Description**
0	bSelector	1	Number	The setting for the attribute of the Selector Control.

5.2.2.4 Feature Unit Control Requests

The following paragraphs describe the Set and Get Feature Unit Control requests. These Feature Unit Control requests can have two forms. The first form must be supported while the second form can be optionally implemented.

5.2.2.4.1 Set Feature Unit Control Request

This request is used to set an attribute of an audio Control inside a Feature Unit of the audio function.

Table 5-14: Set Feature Unit Control Request Values

bmRequest Type	**bRequest**	**wValue**	**wIndex**	**wLength**	**Data**
00100001B	SET_CUR SET_MIN SET_MAX SET_RES	CS and CN	Feature Unit ID and Interface	Length of parameter block	Parameter block

The **bRequest** field indicates which attribute the request is manipulating. The MIN, MAX, and RES attributes are usually not supported for the Set request. Further details on which attributes are supported for which Controls can be found in Section 5.2.2.4.3, “Feature Unit Controls.”

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. (Volume, Mute, etc.) The Channel Number (CN) indicates which logical channel of the cluster is to be influenced. If the request specifies an unknown or unsupported CS or CN to that Unit, the control pipe must indicate a stall.

A special case arises when the Channel Number is set to 0xFF. Then a single Set Feature Unit Control request can be used to set an attribute of all available Controls of a certain type (indicated by the CS) within the Unit. The number of parameters passed in the parameter block must exactly match the number of available Controls in the Unit (as indicated by the number of bits set in the **bmaProps()** array for a certain Control). If this is not the case, the control pipe must indicate a stall. The first parameter in the parameter block is assigned to the attribute of the first available Control, i.e. the one with the lowest Channel Number (including Channel Number 0, the master channel). The above description is referred to as the second form of the Set Feature Unit Control request.

For a description of the parameter blocks for the different Controls that can be addressed through the Set Feature Unit Control request, see Section 5.2.2.4.3, “Feature Unit Controls.”

5.2.2.4.2 Get Feature Unit Control Request

This request returns the attribute setting of a specific audio Control inside a Feature Unit of the audio function.

Table 5-15: Get Feature Unit Control Request Values

bmRequest Type	bRequest	wValue	wIndex	wLength	Data
10100001B	GET_CUR GET_MIN GET_MAX GET_RES	CS and CN	Feature Unit ID and Interface	Length of parameter block	Parameter block

The **bRequest** field indicates which attribute the request is reading.

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 addressing. (Volume, Mute, etc.) The Channel Number (CN) indicates which logical channel of the cluster is to be addressed. If the request specifies an unknown or unsupported CS or CN to that Unit, the control pipe must indicate a stall.

A special case arises when the Channel Number is set to 0xFF. Then a single Get Feature Unit Control request can be used to retrieve the settings of an attribute of all available Controls of a certain type (indicated by the CS) within the Unit. The first parameter returned in the parameter block corresponds to the attribute of the first available Control, i.e. the one with the lowest Channel Number (including Channel Number 0, the master channel). The above description is referred to as the second form of the Get Feature Unit Control request.

For a description of the parameter blocks for the different Controls that can be addressed through the Get Feature Unit Control request, see Section 5.2.2.4.3, “Feature Unit Controls.”

5.2.2.4.3 Feature Unit Controls

The following paragraphs present a detailed description of all possible Controls a Feature Unit can incorporate. For each Control, the layout of the parameter block together with the appropriate Control Selector is listed for all forms of the Get/Set Feature Unit Control request. The Control Selector codes are defined in Section A.10.2, “Feature Unit Control Selectors.”

5.2.2.4.3.1 Mute Control

The Mute Control is one of the building blocks of a Feature Unit. A Mute Control can have only the current setting attribute (CUR). The position of a Mute Control CUR attribute can be either TRUE or FALSE.

In the first form of the request, 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.

Table 5-16: First Form of the Mute Control Parameter Block

Control Selector	MUTE_CONTROL
wLength	1

Offset	Field	Size	Value	Description
0	bMute	1	Bool	The setting for the addressed Mute Control's CUR attribute. Muted when TRUE, not muted when FALSE.

In the second form, the Channel Number field is set to 0xFF. The parameter block contains a list of settings for the CUR attribute for all available Mute Controls in the Feature Unit.

Table 5-17: Second Form of the Mute Control Parameter Block

Control Selector		MUTE_CONTROL		
wLength		Number of available Controls: NrAv		
Offset	**Field**	**Size**	**Value**	**Description**
0	bMute(1)	1	Bool	The setting for the CUR attribute of the first Mute Control.
…	…	…	…	…
NrAv-1	bMute(NrAv)	1	Bool	The setting for the CUR attribute of the last Mute Control.

5.2.2.4.3.2 Volume Control

The Volume Control is one of the building blocks of a Feature Unit. A Volume Control can support all possible Control attributes (CUR, MIN, MAX, and RES). 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 range for the CUR attribute is extended by code 0x8000, representing silence, i.e., -∞ dB. The settings for the RES attribute can only take positive values and range from 1/256 dB (0x0001) to +127.9961 dB (0x7FFF). The Volume Control honors the request to the best of its abilities. It may round the **wVolume** attribute value to its closest available setting. It will report this rounded setting when queried during a Get Control request.

In the first form of the request, 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.

Table 5-18: First Form of the Volume Control Parameter Block

Control Selector	VOLUME_CONTROL
wLength	2

Offset	Field	Size	Value	Description
0	wVolume	2	Number	The setting for the attribute of the addressed Volume Control: 0x7FFF: 127.9961 dB … 0x0100: 1.0000 dB … 0x0002: 0.0078 dB 0x0001: 0.0039 dB 0x0000: 0.0000 dB 0xFFFF: -0.0039 dB 0xFFFE: -0.0078 dB … 0xFE00: -1.0000 dB … 0x8002: -127.9922 dB 0x8001: -127.9961 dB 0x8000 -∞ dB (CUR attribute only)

In the second form, the Channel Number field is set to 0xFF. The parameter block contains a list of settings for an attribute of all available Volume Controls in the Feature Unit.

Table 5-19: Second Form of the Volume Control Parameter Block

Control Selector		VOLUME_CONTROL		
wLength		(Number of available Controls: NrAv)*2		
Offset	**Field**	**Size**	**Value**	**Description**
0	wVolume(1)	1	Number	The setting for the attribute of the first Volume Control.
…	…	…	…	…
(NrAv-1)*2	wVolume(NrAv)	1	Number	The setting for the attribute for the last Volume Control.

5.2.2.4.3.3 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 can support all possible Control attributes (CUR, MIN, MAX, and RES). 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 take positive values and range from 0.25 dB (0x01) to +31.75 dB (0x7F). The Bass Control honors the request to the best of its abilities. It may round the **bBass** attribute value to its closest available setting. It will report this setting when queried during a Get Control request. Other parameters that also influence the behavior of the Bass Control, such as cut-off frequency, cannot be altered through this request.

In the first form of the 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.

Table 5-20: First Form of the Bass Control Parameter Block

Control Selector		BASS_CONTROL		
wLength		1		
Offset	**Field**	**Size**	**Value**	**Description**
0	bBass	1	Number	The setting for the attribute of the addressed Bass Control: 0x7F: +31.75 dB 0x7E: +31.50 dB … 0x00: 0.00 dB … 0x82: -31.50 dB 0x81: -31.75 dB 0x80: -32.00 dB

In the second form, the Channel Number field is set to 0xFF. The parameter block contains a list of settings for all available Bass Controls in the Feature Unit.

Table 5-21: Second Form of the Bass Control Parameter Block

Control Selector		BASS_CONTROL		
wLength		Number of available Controls: NrAv		
Offset	**Field**	**Size**	**Value**	**Description**
0	bBass(1)	1	Number	The setting for the attribute of the first Bass Control.
…	…	…	…	…
NrAv-1	bBass(NrAv)	1	Number	The setting for the attribute of the last Bass Control.

5.2.2.4.3.4 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 can support all possible Control attributes (CUR, MIN, MAX, and RES). 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 take positive values and range from 0.25 dB (0x01) to +31.75 dB (0x7F). The Mid Control honors the request to the best of its abilities. It may round the **bMid** attribute value to its closest available setting. It will report this setting when queried during a Get Audio Control request. Other parameters that also influence the behavior of the Mid Control, such as cut-off frequency, cannot be altered through this request.

In the first form of the 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.

Table 5-22: First Form of the Mid Control Parameter Block

Control Selector		MID_CONTROL		
wLength		1		
Offset	**Field**	**Size**	**Value**	**Description**
0	bMid	1	Number	The setting for the attribute of the addressed Mid Control: 0x7F: +31.75 dB 0x7E: +31.50 dB … 0x00: 0.00 dB … 0x82: -31.50 dB 0x81: -31.75 dB 0x80: -32.00 dB

In the second form, the Channel Number field is set to 0xFF. The parameter block contains a list of settings for all available Mid Controls in the Feature Unit.

Table 5-23: Second Form of the Mid Control Parameter Block

Control Selector		MID_CONTROL		
wLength		Number of available Controls: NrAv		
Offset	**Field**	**Size**	**Value**	**Description**
0	bMid(1)	1	Number	The setting for the attribute of the first Mid Control.
…	…	…	…	…
NrAv-1	bMid(NrAv)	1	Number	The setting for the attribute of the last Mid Control.

5.2.2.4.3.5 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 can support all possible Control attributes (CUR, MIN, MAX, and RES). 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 take positive values and range from 0.25 dB (0x01) to +31.75 dB (0x7F). The Treble Control honors the request to the best of its abilities. It may round the **bTreble** attribute value to its closest available setting. It will report this setting when queried during a Get Control request. Other parameters that also influence the behavior of the Treble Control, such as cut-off frequency, cannot be altered through this request.

In the first form of the 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.

Table 5-24: First Form of the Treble Control Parameter Block

Control Selector		TREBLE_CONTROL		
wLength		1		
Offset	**Field**	**Size**	**Value**	**Description**
0	bTreble	1	Number	The setting for the attribute of the addressed Treble Control: 0x7F: +31.75 dB 0x7E: +31.50 dB … 0x00: 0.00 dB … 0x82: -31.50 dB 0x81: -31.75 dB 0x80: -32.00 dB

In the second form, the Channel Number field is set to 0xFF. The parameter block contains a list of settings for all available Treble Controls in the Feature Unit.

Table 5-25: Second Form of the Treble Control Parameter Block

Control Selector		TREBLE_CONTROL		
wLength		Number of available Controls: NrAv		
Offset	**Field**	**Size**	**Value**	**Description**
0	bTreble(1)	1	Number	The setting for the attribute of the first Treble Control.
…	…	…	…	…
NrAv-1	bTreble(NrAv)	1	Number	The setting for the attribute of the last Treble Control.

5.2.2.4.3.6 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 5-26: 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

Band Nr.	Center Freq.	Band Nr.	Center Freq.	Band Nr.	Center Freq.
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 can support all possible Control attributes (CUR, MIN, MAX, and RES). However, if a certain attribute is supported, it must do so for all individual bands. 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 take positive values and range from 0.25 dB (0x01) to +31.75 dB (0x7F). The Graphic Equalizer Control honors the request to the best of its abilities. It may round the **bBandxx** attribute values to their closest available settings. It will report these settings when queried during a Get Control request.

Only the first form of the Feature Unit Control request for Graphic Equalizer Control is supported. 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.

Table 5-27: Graphic Equalizer Control Parameter Block

Control Selector	GRAPHIC_EQUALIZER_CONTROL
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	bBand(Lowest)	1	Number	The setting for the 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)	bBand(Highest)	1	Number	The setting for the attribute of the highest band present.

5.2.2.4.3.7 Automatic Gain Control

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

In the first form of the request, 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.

Table 5-28: First Form of the Automatic Gain Control Parameter Block

Control Selector		AUTOMATIC_GAIN_CONTROL		
wLength		1		
Offset	**Field**	**Size**	**Value**	**Description**
0	bAGC	1	Bool	The setting for the attribute of the addressed Automatic Gain Control. On when TRUE, off when FALSE.

In the second form, the Channel Number field is set to 0xFF. The parameter block contains a list of settings for all available AGC Controls in the Feature Unit.

Table 5-29: Second Form of the Automatic Gain Control Parameter Block

Control Selector	AUTOMATIC_GAIN_CONTROL

wLength		Number of available Controls: NrAv		
Offset	**Field**	**Size**	**Value**	**Description**
0	bAGC(1)	1	Bool	The setting for the attribute of the first Automatic Gain Control.
…	…	…	…	…
NrAv-1	bAGC(NrAv)	1	Bool	The setting for the attribute of the last Automatic Gain Control.

5.2.2.4.3.8 Delay Control

The Delay Control is one of the building blocks of a Feature Unit. A Delay Control can support all possible Control attributes (CUR, MIN, MAX, and RES). The settings for the CUR, MIN, MAX, and RES attributes can range from zero (0x0000) to 1023.9844ms (0xFFFF) in steps of 1/64 ms or 0.015625 ms (0x0001). The Delay Control honors the request to the best of its abilities. It may round the **wDelay** attribute value to its closest available setting. It will report this rounded setting when queried during a Get Control request.

In the first form of the request, 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.

Table 5-30: First Form of the Delay Control Parameter Block

Control Selector		DELAY_CONTROL		
wLength		2		
Offset	**Field**	**Size**	**Value**	**Description**
0	wDelay	2	Number	The setting for the attribute of the addressed Delay Control: 0x0000: 0.0000 ms 0x0001: 0.0156 ms 0x0002: 0.0312 ms … 0x0040: 1.0000 ms … 0xFFFD: 1023.9531 ms 0xFFFE: 1023.9687 ms 0xFFFF 1023.9844 ms

In the second form, the Channel Number field is set to 0xFF. The parameter block contains a list of settings for all available Delay Controls in the Feature Unit.

Table 5-31: Second Form of the Delay Control Parameter Block

Control Selector	DELAY_CONTROL
wLength	(Number of available Controls: NrAv)*2

Offset	Field	Size	Value	Description
0	wDelay(1)	1	Number	The setting for the attribute of the first Delay Control.
…	…	…	…	…
(NrAv-1)*2	wDelay(NrAv)	1	Number	The setting for the attribute of the last Delay Control.

5.2.2.4.3.9 Bass Boost Control

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

In the first form of the request, 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.

Table 5-32: First Form of the Bass Boost Control Parameter Block

Control Selector		BASS_BOOST_CONTROL		
wLength		1		
Offset	**Field**	**Size**	**Value**	**Description**
0	bBassBoost	1	Bool	The setting for the addressed Bass Boost Control’s CUR attribute. On when TRUE, off when FALSE.

In the second form, the Channel Number field is set to 0xFF. The parameter block contains a list of settings for the CUR attribute for all available Bass Boost Controls in the Feature Unit.

Table 5-33: Second Form of the Bass Boost Control Parameter Block

Control Selector		BASS_BOOST_CONTROL		
wLength		Number of available Controls: NrAv		
Offset	**Field**	**Size**	**Value**	**Description**
0	bBassBoost(1)	1	Bool	The setting for the CUR attribute of the first Bass Boost Control.
…	…	…	…	…
NrAv-1	bBassBoost(NrAv)	1	Bool	The setting for the CUR attribute of the last Bass Boost Control.

5.2.2.4.3.10 Loudness Control

The Loudness Control is one of the building blocks of a Feature Unit. A Loudness Control can have only the current setting attribute (CUR). The position of a Loudness Control CUR attribute can be either TRUE or FALSE.

In the first form of the request, 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.

Table 5-34: First Form of the Loudness Control Parameter Block

Control Selector		LOUDNESS_CONTROL		
wLength		1		
Offset	**Field**	**Size**	**Value**	**Description**
0	bLoudness	1	Bool	The setting for the addressed Loudness Control's CUR attribute. On when TRUE, off when FALSE.

In the second form, the Channel Number field is set to 0xFF. The parameter block contains a list of settings for the CUR attribute for all available Loudness Controls in the Feature Unit.

Table 5-35: Second Form of the Loudness Control Parameter Block

Control Selector		LOUDNESS_CONTROL		
wLength		Number of available Controls: NrAv		
Offset	**Field**	**Size**	**Value**	**Description**
0	bLoudness(1)	1	Bool	The setting for the CUR attribute of the first Loudness Control.
…	…	…	…	…
NrAv-1	bLoudness(NrAv)	1	Bool	The setting for the CUR attribute of the last Loudness Control.

5.2.2.5 Processing Unit Control Requests

The following sections describe the Set and Get Processing Unit Control requests. They are used to manipulate the audio Controls within a Processing Unit.

5.2.2.5.1 Set Processing Unit Control Request

This request is used to set an attribute of an audio Control inside a Processing Unit of the audio function.

Table 5-36: Set Processing Unit Control Request Values

bmRequest Type	bRequest	wValue	wIndex	wLength	Data
00100001B	SET_CUR SET_MIN SET_MAX SET_RES	CS	Processing Unit ID and Interface	Length of parameter block	Parameter block

The **bRequest** field indicates which attribute the request is manipulating. The MIN, MAX, and RES attributes are usually not supported for the Set request.

The **wValue** field specifies the Control Selector (CS) in the high byte and zero in the low byte. The Control Selector indicates which type of Control this request is manipulating (Enable Processing, Mode Select, etc.) If the request specifies an unknown or unsupported CS to that Processing Unit, the control pipe must indicate a stall.

For a description of the parameter block for the Processing Unit Controls, see Section 5.2.2.5.3, “Processing Unit Controls.”

5.2.2.5.2 Get Processing Unit Control Request

This request returns the attribute setting of a specific audio Control inside a Processing Unit of the audio function.

Table 5-37: Get Processing Unit Control Request Values

bmRequest Type	bRequest	wValue	wIndex	wLength	Data
10100001B	GET_CUR GET_MIN GET_MAX GET_RES	CS	Unit ID and Interface	Length of parameter block	Parameter block

The **bRequest** field indicates which attribute the request is reading.

The **wValue** field specifies the Control Selector (CS) in the high byte and zero in the low byte. The Control Selector indicates which type of Control this request is addressing. (Enable Processing, Mode Select, etc.) If the request specifies an unknown or unsupported CS to that Unit, the control pipe must indicate a stall.

For a description of the parameter block for the Processing Unit Controls, see Section 5.2.2.5.3, “Processing Unit Controls.”

5.2.2.5.3 Processing Unit Controls

Processing Unit Controls are in principle specific to the type of process a Processing Unit implements. However, instead of specifying all supported Controls on a per process basis, it was considered more efficient to create a ‘pool’ of possible Controls. In this way, Controls that occur in different Processing Units need only be specified once. However, the Control Selector Codes and the bit positions in the **bmControls** field of the Processing Unit descriptor are assigned on a per-process type basis. Issuing non-supported Control Selectors to a Processing Unit leads to a control pipe stall.

The following paragraphs present a detailed description of all possible Controls a Processing Unit can incorporate. For each Control, the layout of the parameter block together with the appropriate Control Selector is listed for all forms of the Get/Set Processing Unit Control request. The Control Selector codes are defined in Section A.10.3, “Processing Unit Control Selectors.”

5.2.2.5.3.1 Enable Processing Control

The Enable Processing Control is used to either enable the functionality of the Processing Unit or bypass the Processing Unit entirely. In the latter case, default behavior is assumed. Enable Processing Control can have only the current setting attribute (CUR). The position of an Enable Processing switch can be either TRUE or FALSE. The current setting of the Control can be queried using a Get Processing Unit Control request.

Table 5-38: Enable Processing Control Parameter Block

Control Selector		XX_ENABLE_CONTROL		
wLength		1		
Offset	**Field**	**Size**	**Value**	**Description**
0	bEnable	1	Bool	The setting for the Enable Processing Control CUR attribute. Enabled when TRUE, disabled when FALSE.

5.2.2.5.3.2 Mode Select Control

The Mode Select Control is used to change the behavior of the Processing Unit. A Mode Select Control can support all possible Control attributes (CUR, MIN, MAX, and RES).The valid range for the CUR, MIN, and MAX attributes is from one to the number of modes, supported by the Processing Unit (reported through the **bNrModes** field of the Processing Unit descriptor). The RES attribute can only have a value of one. The Mode Select Control honors the request to the best of its abilities. It may round the **bMode** attribute value to its closest available setting. It will report this rounded setting when queried during a Get Processing Unit Control request.

Table 5-39: Mode Select Control Parameter Block

Control Selector		XX_MODE_SELECT_CONTROL		
wLength		1		
Offset	**Field**	**Size**	**Value**	**Description**
0	bMode	1	Number	The setting for the attribute of the Mode Select Control.

5.2.2.5.3.3 Spaciousness Control

The Spaciousness Control is used to change the spatial appearance of the stereo image, produced by the 3D-Stereo Extender Processing Unit. The Spaciousness Control can support all possible Control attributes (CUR, MIN, MAX, and RES). The valid range for the CUR, MIN, MAX, and RES attributes is from zero to 255%. The Spaciousness Control honors the request to the best of its abilities. It may round the **bSpaciousness** attribute value to its closest available setting. It will report this rounded setting when queried during a Get Processing Unit Control request.

Table 5-40: Spaciousness Control Parameter Block

Control Selector		SPACIOUSNESS_CONTROL		
wLength		1		
Offset	**Field**	**Size**	**Value**	**Description**
0	bSpaciousness	1	Number	The setting for the attribute of the Spaciousness Control.

5.2.2.5.3.4 Reverb Type Control

The Reverb Type Control is a macro parameter that allows global settings of reverb parameters within the Reverberation Processing Unit. When a certain Reverb Type is selected, each reverb parameter will be set to the most suitable value. The Reverb Type Control can support all possible Control attributes (CUR, MIN, MAX, and RES). 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 Reverb Type Control honors the request to the best of its abilities. It may round the **bReverbType** attribute value to its closest available setting. It will report this rounded setting when queried during a Get Processing Unit Control request.

Table 5-41: Reverb Type Control Parameter Block

Control Selector		REVERB_TYPE_CONTROL		
wLength		1		
Offset	**Field**	**Size**	**Value**	**Description**
0	bReverbType	1	Number	The setting for the attribute of the Reverb Type Control.

5.2.2.5.3.5 Reverb Level Control

The Reverb Level Control is used to set the amount of reverberant sound introduced by the Reverberation Processing Unit. The Reverb Level Control can support all possible Control attributes (CUR, MIN, MAX ,and RES). 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 Reverb Level Control honors the request to the best of its abilities. It may round the **bReverbLevel** attribute value to its closest available setting. It will report this rounded setting when queried during a Get Processing Unit Control request.

Table 5-42: Reverb Level Control Parameter Block

Control Selector		REVERB_LEVEL_CONTROL		
wLength		1		
Offset	**Field**	**Size**	**Value**	**Description**
0	bReverbLevel	1	Number	The setting for the attribute of the Reverb Level Control.

5.2.2.5.3.6 Reverb Time Control

The Reverb Time Control is used to set the time over which the reverberation, introduced by the Reverberation Processing Unit, will continue. The Reverb Time Control can support all possible Control attributes (CUR, MIN, MAX, and RES). 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 Reverb Time Control honors the request to the best of its abilities. It may round the **wReverbTime** attribute value to its closest available setting. It will report this rounded setting when queried during a Get Processing Unit Control request.

Table 5-43: Spaciousness Control Parameter Block

Control Selector		REVERB_TIME_CONTROL		
wLength		2		
Offset	**Field**	**Size**	**Value**	**Description**
0	wReverbTime	2	Number	The setting for the attribute of the Reverb Time Control.

5.2.2.5.3.7 Reverb Delay Feedback Control

The Reverb Delay Feedback Control is used when the Reverb Type is set to Reverb Type 6, Delay or Reverb Type 7, Panning Delay. It sets the way in which delay repeats. The Reverb Delay Feedback Control range can support all possible Control attributes (CUR, MIN, MAX, and RES). 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 Reverb Delay Feedback Control honors the request to the best of its abilities. It may round the **bReverbFeedback** attribute value to its closest available setting. It will report this rounded setting when queried during a Get Processing Unit Control request.

Table 5-44: Reverb Delay Feedback Control Parameter Block

Control Selector	REVERB_FEEDBACK_CONTROL
wLength	1

Offset	Field	Size	Value	Description
0	bReverbFeedback	1	Number	The setting for the attribute of the Reverb Delay Feedback Control.

5.2.2.5.3.8 Chorus Level Control

The Chorus Level Control is used to set the amount of chorus effect sound introduced by the Chorus Processing Unit. The Chorus Level Control can support all possible Control attributes (CUR, MIN, MAX, and RES). 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 Chorus Level Control honors the request to the best of its abilities. It may round the **bChorusLevel** attribute value to its closest available setting. It will report this rounded setting when queried during a Get Processing Unit Control request.

Table 5-45: Chorus Level Control Parameter Block

Control Selector		CHORUS_LEVEL_CONTROL		
wLength		1		
Offset	**Field**	**Size**	**Value**	**Description**
0	bChorusLevel	1	Number	The setting for the attribute of the Chorus Level Control.

5.2.2.5.3.9 Chorus Modulation Rate Control

The Chorus Modulation Rate Control is used to set the speed (frequency) of the modulator of the chorus, introduced by the Chorus Processing Unit. Higher values result in faster modulation. The Chorus Modulation Rate Control can support all possible Control attributes (CUR, MIN, MAX, and RES). 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 Chorus Modulation Rate Control honors the request to the best of its abilities. It may round the **wChorusRate** attribute value to its closest available setting. It will report this rounded setting when queried during a Get Processing Unit Control request.

Table 5-46: Chorus Modulation Rate Control Parameter Block

Control Selector		CHORUS_RATE_CONTROL		
wLength		2		
Offset	**Field**	**Size**	**Value**	**Description**
0	wChorusRate	2	Number	The setting for the attribute of the Chorus Modulation Rate Control.

5.2.2.5.3.10 Chorus Modulation Depth Control

The Chorus Modulation Depth Control is used to set the depth at which the chorus sound introduced by the Chorus Processing Unit, is modulated. Higher values result in deeper modulation. The Chorus Modulation Depth Control can support all possible Control attributes (CUR, MIN, MAX, and RES). 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 Chorus Modulation Depth Control honors the request to the best of its abilities. It may round the **wChorusDepth** attribute value to its closest

available setting. It will report this rounded setting when queried during a Get Processing Unit Control request.

Table 5-47: Chorus Modulation Depth Control Parameter Block

Control Selector		CHORUS_DEPTH_CONTROL		
wLength		2		
Offset	**Field**	**Size**	**Value**	**Description**
0	wChorusDepth	2	Number	The setting for the attribute of the Chorus Modulation Depth Control.

5.2.2.5.3.11 Dynamic Range Compressor 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 can support all possible Control attributes (CUR, MIN, MAX, and RES). The valid range for the CUR, MIN, MAX, and RES attributes is from zero (0x0000) to 255.9961 dB/dB (0xFFFF) in steps of 1/256 dB/dB or 0.00390625 dB/dB (0x0001). The Compression Ratio Control honors the request to the best of its abilities. It may round the **wRatio** attribute value to its closest available setting. It will report this rounded setting when queried during a Get Processing Unit Control request.

Table 5-48: Dynamic Range Compressor Ratio Control Parameter Block

Control Selector		COMPRESSION_RATIO_CONTROL		
wLength		2		
Offset	**Field**	**Size**	**Value**	**Description**
0	wRatio	2	Number	The setting for the attribute of the Compression Ratio Control.

5.2.2.5.3.12 Dynamic Range Compressor MaxAmpl Control

The MaxAmpl Control is used to set the upper boundary of the active input range of the compressor. The MaxAmpl Control can support all possible Control attributes (CUR, MIN, MAX, and RES). 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 take positive values and range from 1/256 dB (0x0001) to +127.9961 dB (0x7FFF). The MaxAmpl Control honors the request to the best of its abilities. It may round the **wMaxAmpl** attribute value to its closest available setting. It will report this rounded setting when queried during a Get Processing Unit Control request.

Table 5-49: Dynamic Range Compressor MaxAmpl Control Parameter Block

Control Selector	MAXAMPL_CONTROL
wLength	2

Offset	Field	Size	Value	Description
0	wMaxAmpl	2	Number	The setting for the attribute of the MaxAmpl Control.

5.2.2.5.3.13 Dynamic Range Compressor Threshold Control

The Threshold Control is used to set the lower boundary of the active input range of the compressor. The Threshold Control can support all possible Control attributes (CUR, MIN, MAX, and RES). 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 take positive values and range from 1/256 dB (0x0001) to +127.9961 dB (0x7FFF). The Threshold Control honors the request to the best of its abilities. It may round the **wThreshol** attribute value to its closest available setting. It will report this rounded setting when queried during a Get Processing Unit Control request.

Table 5-50: Dynamic Range Compressor Threshold Control Parameter Block

Control Selector		THRESHOLD_CONTROL		
wLength		2		
Offset	**Field**	**Size**	**Value**	**Description**
0	wThreshold	2	Number	The setting for the attribute of the Threshold Control.

5.2.2.5.3.14 Dynamic Range Compressor 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 can support all possible Control attributes (CUR, MIN, MAX, and RES). 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 Attack Time Control honors the request to the best of its abilities. It may round the **wAttackTime** attribute value to its closest available setting. It will report this rounded setting when queried during a Get Processing Unit Control request.

Table 5-51: Dynamic Range Compressor Attack Time Control Parameter Block

Control Selector		ATTACK_TIME_CONTROL		
wLength		2		
Offset	**Field**	**Size**	**Value**	**Description**
0	wAttackTime	2	Number	The setting for the attribute of the Attack Time Control.

5.2.2.5.3.15 Dynamic Range Compressor 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 can support all possible Control attributes (CUR, MIN, MAX, and RES). 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 Release Time Control honors the request to the best of its abilities. It may round the **wReleaseTime** attribute value to its

closest available setting. It will report this rounded setting when queried during a Get Processing Unit Control request.

Table 5-52: Dynamic Range Compressor Release Time Control Parameter Block

Control Selector		RELEASE_TIME_CONTROL		
wLength		2		
Offset	**Field**	**Size**	**Value**	**Description**
0	wAttackTime	2	Number	The setting for the attribute of the Release Time Control.

5.2.2.6 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.

5.2.2.6.1 Set Extension Unit Control Request

This request is used to set an audio Control inside a particular Extension Unit of the audio function.

Table 5-53: Set Extension Unit Control Request Values

bmRequest Type	**bRequest**	**wValue**	**wIndex**	**wLength**	**Data**
00100001B	SET_CUR SET_MIN SET_MAX SET_RES	CS	Extension Unit ID and Interface	Length of parameter block	Parameter block

The **bRequest** field indicates which attribute the request is manipulating. The MIN, MAX, and RES attributes are usually not supported for the Set request.

The **wValue** field specifies the Control Selector (CS) in the high byte and zero in the low byte. The Control Selector indicates which type of Control this request is manipulating. (Enable Processing, etc.) If the request specifies an unknown or unsupported CS to that Unit, the control pipe must indicate a stall.

For a description of the parameter block for the Extension Unit Control Selectors, see Section 5.2.2.6.3, “Extension Unit Controls.”

5.2.2.6.2 Get Extension Unit Control Request

This request returns the attribute setting of a specific audio Control inside an Extension Unit of the audio function.

Table 5-54: Get Extension Unit Control Request Values

bmRequest Type	bRequest	wValue	wIndex	wLength	Data
10100001B	GET_CUR GET_MIN GET_MAX GET_RES	CS	Extension Unit ID and Interface	Length of parameter block	Parameter block

The **bRequest** field indicates which attribute the request is reading

The **wValue** field specifies the Control Selector (CS) in the high byte and zero in the low byte. The Control Selector indicates which type of Control this request is addressing. (Enable Processing, etc.) If the request specifies an unknown or unsupported CS to that Unit, the control pipe must indicate a stall.

For a description of the parameter block for the Extension Unit Controls, see Section 5.2.2.6.3, “Extension Unit Controls.”

5.2.2.6.3 Extension Unit Controls

For the time being, this specification only defines a single generic Extension Unit Control, the Enable Processing Control. Issuing non-supported Control Selectors to an Extension Unit leads to a control pipe stall.

5.2.2.6.3.1 Enable Processing Control

The Enable Processing Control is used to either enable the functionality of the Extension Unit or bypass the Extension Unit entirely. In the latter case, default behavior is assumed. Enable Processing Control can have only the current setting attribute (CUR). The position of an Enable Processing switch can be either TRUE or FALSE. The current setting of the Control can be queried using a Get Extension Unit Control request.

Table 5-55: Enable Processing Control Parameter Block

Control Selector		XU_ENABLE_CONTROL		
wLength		1		
Offset	**Field**	**Size**	**Value**	**Description**
0	bOn	1	Bool	The setting for the Enable Processing Control. Enabled when TRUE, disabled when FALSE.

5.2.3 AudioStreaming 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.

5.2.3.1 Interface Control Requests

For now, this specification only deals with format-specific AudioStreaming interface requests. Depending on the Audio Data Format used by the interface (reflected in the **wFormatTag** field of the class-specific AS interface descriptor), the set of requests to control the interface 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 yields 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. Therefore, the format-specific Interface Control requests are described in the *USB Audio Data Formats*, together with the definition of the format-specific descriptors.

5.2.3.2 Endpoint Control Requests

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

5.2.3.2.1 Set Endpoint Control Request

This request is used to set an attribute of an endpoint Control inside a particular endpoint of the audio function.

Table 5-56: Set Endpoint Control Request Values

bmRequest Type	bRequest	wValue	wIndex	wLength	Data
00100010B	SET_CUR SET_MIN SET_MAX SET_RES	CS	endpoint	Length of parameter block	Parameter block

The **bRequest** field indicates which attribute the request is manipulating. The MIN, MAX, and RES attributes are usually not supported for the Set request.

The **wValue** field specifies the Control Selector (CS) in the high byte and the low byte must be set to zero. The Control Selector indicates which type of Control this request is manipulating (Sampling Frequency, Pitch Control, etc.) If the request specifies an unknown CS to that endpoint, the control pipe must indicate a stall.

For a description of the parameter block for the endpoint Control Selectors, see Section 5.2.3.2.3, “Endpoint Controls.”

5.2.3.2.2 Get Endpoint Control Request

This request returns the attribute setting of a specific endpoint Control inside an endpoint of the audio function.

Table 5-57: Get Endpoint Control Request Values

bmRequest Type	bRequest	wValue	wIndex	wLength	Data
10100010B	GET_CUR GET_MIN GET_MAX GET_RES	CS	endpoint	Length of parameter block	Parameter block

The **bRequest** field indicates which attribute the request is reading.

The **wValue** field specifies the Control Selector (CS) in the high byte and the low byte must be set to zero. The Control Selector indicates which type of Control this request is manipulating (Sampling Frequency,

Pitch Control, etc.) If the request specifies an unknown CS to that endpoint, the control pipe must indicate a stall.

For a description of the parameter block for the endpoint Control Selectors, see Section 5.2.3.2.3, “Endpoint Controls.”

5.2.3.2.3 Endpoint Controls

5.2.3.2.3.1 Sampling Frequency Control

The Sampling Frequency Control is used to set the initial sampling frequency for an isochronous audio data endpoint. This allows the endpoints’ clock recovery system to lock onto the incoming clock much faster. Adaptive endpoints can benefit from this. The Sampling Frequency Control can support all possible Control attributes (CUR, MIN, MAX, and RES). The settings for the CUR, MIN, and MAX attributes can range from 0 Hz (0x000000) to 8388607 Hz (0x7FFFFF) in steps of 1 Hz (0x0001). The Sampling Frequency Control honors the request to the best of its abilities. If the endpoint operates at a fixed sampling frequency, setting this Control has no effect. If the endpoint supports a discrete number of sampling frequencies, setting the **tSampleFreq** value to a non-supported value causes the Control to round it to the closest available value. This also happens when the sampling frequency is set outside the range for a continuous sampling frequency endpoint. It will report the rounded setting when queried during a Get Control request.

Table 5-58: Sampling Frequency Control Parameter Block

Control Selector		SAMPLING_FREQ_CONTROL		
wLength		3		
Offset	**Field**	**Size**	**Value**	**Description**
0	tSampleFreq	3	Number	The sampling frequency expressed in Hz.

5.2.3.2.3.2 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 can have only the current setting attribute (CUR). The position of a Pitch Control CUR attribute can be either TRUE or FALSE.

Table 5-59: Pitch Control Parameter Block

Control Selector		PITCH_CONTROL		
wLength		1		
Offset	**Field**	**Size**	**Value**	**Description**
0	bPitchEnable	1	Bool	Pitch Control on when TRUE, off when FALSE.

5.2.4 Additional Requests

5.2.4.1 Memory Requests

The Host can interact with an addressable Entity (Terminal, Unit 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.

5.2.4.1.1 Set Memory Request

This request is used to download a parameter block into a particular Entity of the audio function.

Table 5-60: Set Memory Request Values

bmRequest Type	bRequest	wValue	wIndex	wLength	Data
00100001B	SET_MEM	Offset	Entity ID and Interface	Length of parameter block	Parameter block
00100010B			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 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.

5.2.4.1.2 Get Memory Request

This request is used to upload a parameter block from a particular Entity of the audio function.

Table 5-61: Get Memory Request Values

bmRequest Type	bRequest	wValue	wIndex	wLength	Data
10100001B	GET_MEM	Offset	Entity ID and Interface	Length of parameter block	Parameter block
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 parameter space.

The layout of the parameter block is implementation dependent.

5.2.4.2 Get Status Request

This request is used to retrieve status information from an Entity within the audio function.

Table 5-62: Get Status Request Values

bmRequest Type	bRequest	wValue	wIndex	wLength	Data
10100001B	GET_STAT	Zero	Entity ID and Interface	Status message length	Status message
10100010B			Endpoint		

The **bRequest** field contains the GET_STAT constant, defined in Section A.9, “Audio Class-Specific Request Codes.”

The **wValue** field is currently unused and must be set to zero.

The **wLength** field specifies the number of bytes to return. If the status message is longer than the **wLength** field, only the initial bytes of the status message are returned. If the status message is shorter than the **wLength** field, the function indicates the end of the control transfer by sending short packet when further data is requested.

The contents of the status message is reserved for future use. For the time being, a null packet should be returned in the data stage of the control transfer.

Appendix A. Audio Device Class Codes

A.1 Audio Interface Class Code

Table A-1: Audio Interface Class Code

Audio Interface Class Code	Value
AUDIO	0x01

A.2 Audio Interface Subclass Codes

Table A-2: Audio Interface Subclass Codes

Audio Subclass Code	Value
SUBCLASS_UNDEFINED	0x00
AUDIOCONTROL	0x01
AUDIOSTREAMING	0x02
MIDISTREAMING	0x03

A.3 Audio Interface Protocol Codes

Table A-3: Audio Interface Protocol Codes

Audio Protocol Code	Value
PR_PROTOCOL_UNDEFINED	0x00

A.4 Audio Class-Specific Descriptor Types

Table A-4: Audio Class-specific Descriptor Types

Descriptor Type	Value
CS_UNDEFINED	0x20
CS_DEVICE	0x21
CS_CONFIGURATION	0x22
CS_STRING	0x23
CS_INTERFACE	0x24
CS_ENDPOINT	0x25

A.5 Audio Class-Specific AC Interface Descriptor Subtypes

Table A-5: 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
PROCESSING_UNIT	0x07
EXTENSION_UNIT	0x08

A.6 Audio Class-Specific AS Interface Descriptor Subtypes

Table A-6: Audio Class-Specific AS Interface Descriptor Subtypes

Descriptor Subtype	Value
AS_DESCRIPTOR_UNDEFINED	0x00
AS_GENERAL	0x01
FORMAT_TYPE	0x02
FORMAT_SPECIFIC	0x03

A.7 Processing Unit Process Types

Table A-7: Processing Unit Process Types

wProcessType	Value
PROCESS_UNDEFINED	0x00
UP/DOWNMIX_PROCESS	0x01
DOLBY_PROLOGIC_PROCESS	0x02
3D_STEREO_EXTENDER_PROCESS	0x03
REVERBERATION_PROCESS	0x04

wProcessType	Value
CHORUS_PROCESS	0x05
DYN_RANGE_COMP_PROCESS	0x06

A.8 Audio Class-Specific Endpoint Descriptor Subtypes

Table A-8: Audio Class-Specific Endpoint Descriptor Subtypes

Descriptor Subtype	Value
DESCRIPTOR_UNDEFINED	0x00
EP_GENERAL	0x01

A.9 Audio Class-Specific Request Codes

Table A-9: Audio Class-Specific Request Codes

Class-Specific Request Code	Value
REQUEST_CODE_UNDEFINED	0x00
SET_ CUR	0x01
GET_ CUR	0x81
SET_ MIN	0x02
GET_ MIN	0x82
SET_ MAX	0x03
GET_ MAX	0x83
SET_ RES	0x04
GET_ RES	0x84
SET_MEM	0x05
GET_ MEM	0x85
GET_STAT	0xFF

A.10 Control Selector Codes

A.10.1 Terminal Control Selectors

Table A-10: Terminal Control Selectors

Control Selector	Value
TE_CONTROL_UNDEFINED	0x00
COPY_PROTECT_CONTROL	0x01

A.10.2 Feature Unit Control Selectors

Table A-11: Feature Unit Control Selectors

Control Selector	Value
FU_CONTROL_UNDEFINED	0x00
MUTE_CONTROL	0x01
VOLUME_CONTROL	0x02
BASS_CONTROL	0x03
MID_CONTROL	0x04
TREBLE_CONTROL	0x05
GRAPHIC_EQUALIZER_CONTROL	0x06
AUTOMATIC_GAIN_CONTROL	0x07
DELAY_CONTROL	0x08
BASS_BOOST_CONTROL	0x09
LOUDNESS_CONTROL	0x0A

A.10.3 Processing Unit Control Selectors

A.10.3.1 Up/Down-mix Processing Unit Control Selectors

Table A-12: Up/Down-mix Processing Unit Control Selectors

Control Selector	Value
UD_CONTROL_UNDEFINED	0x00
UD_ENABLE_CONTROL	0x01
UD_MODE_SELECT_CONTROL	0x02

A.10.3.2 Dolby Prologic™ Processing Unit Control Selectors

Table A-13: Dolby Prologic Processing Unit Control Selectors

Control Selector	Value
DP_CONTROL_UNDEFINED	0x00
DP_ENABLE_CONTROL	0x01
DP_MODE_SELECT_CONTROL	0x02

A.10.3.3 3D Stereo Extender Processing Unit Control Selectors

Table A-14: 3D Stereo Extender Processing Unit Control Selectors

Control Selector	Value
3D_CONTROL_UNDEFINED	0x00
3D_ENABLE_CONTROL	0x01
SPACIOUSNESS_CONTROL	0x03

A.10.3.4 Reverberation Processing Unit Control Selectors

Table A-15: Reverberation Processing Unit Control Selectors

Control Selector	Value
RV_CONTROL_UNDEFINED	0x00
RV_ENABLE_CONTROL	0x01
REVERB_LEVEL_CONTROL	0x02
REVERB_TIME_CONTROL	0x03
REVERB_FEEDBACK_CONTROL	0x04

A.10.3.5 Chorus Processing Unit Control Selectors

Table A-16: Chorus Processing Unit Control Selectors

Control Selector	Value
CH_CONTROL_UNDEFINED	0x00
CH_ENABLE_CONTROL	0x01
CHORUS_LEVEL_CONTROL	0x02
CHORUS_RATE_CONTROL	0x03

Control Selector	Value
CHORUS_DEPTH_CONTROL	0x04

A.10.3.6 Dynamic Range Compressor Processing Unit Control Selectors

Table A-17: Dynamic Range Compressor Processing Unit Control Selectors

Control Selector	Value
DR_CONTROL_UNDEFINED	0x00
DR_ENABLE_CONTROL	0x01
COMPRESSION_RATE_CONTROL	0x02
MAXAMPL_CONTROL	0x03
THRESHOLD_CONTROL	0x04
ATTACK_TIME	0x05
RELEASE_TIME	0x06

A.10.4 Extension Unit Control Selectors

Table A-18: Extension Unit Control Selectors

Control Selector	Value
XU_CONTROL_UNDEFINED	0x00
XU_ENABLE_CONTROL	0x01

A.10.5 Endpoint Control Selectors

Table A-19: Endpoint Control Selectors

Control Selector	Value
EP_CONTROL_UNDEFINED	0x00
SAMPLING_FREQ_CONTROL	0x01
PITCH_CONTROL	0x02

Appendix B. Example 1: USB Microphone (Informative)

B.1 Product Description

The device described here is a USB Microphone. It is a very simple device that has no Audio Controls incorporated. It delivers a mono audio data stream to the Host over its AudioStreaming interface. The used Audio Data Format is 16-bit 8KHz PCM. The synchronization type is Asynchronous Source. It uses its internal clock as a reference. The following figure presents the internal topology of the microphone.

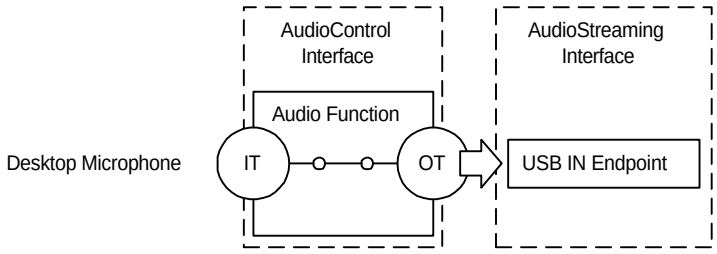

Figure B-1: USB Microphone Topology

The audio function contains one Input Terminal that represents the actual microphone pick-up element, followed by the Analog-to-Digital Converter (ADC). The digital output stream of the ADC enters the audio function through the single Output Pin of the Input Terminal. Because there is no further processing on the audio stream, the Input Terminal is directly connected to the Input Pin of the Output Terminal. The Output Terminal is the representation within the audio function of the USB IN endpoint that eventually delivers the audio data stream to the Host. The internals of the audio function are presented to the Host through the (mandatory) AudioControl interface whereas the USB IN endpoint resides in the AudioStreaming interface.

B.2 Descriptor Hierarchy

This USB Microphone device includes the AudioControl interface (interface 0) and a single AudioStreaming interface (interface 1). The AudioStreaming interface features two alternate settings. The first alternate setting (Alternate Setting 0) has zero bandwidth associated with it so that switching to this alternate setting effectively frees all allocated bandwidth on the USB for this device. Zero bandwidth is indicated by the lack of a streaming endpoint. Alternate Setting 1 is the operational part of the interface and it has one isochronous IN endpoint. Figure presents the descriptor hierarchy.

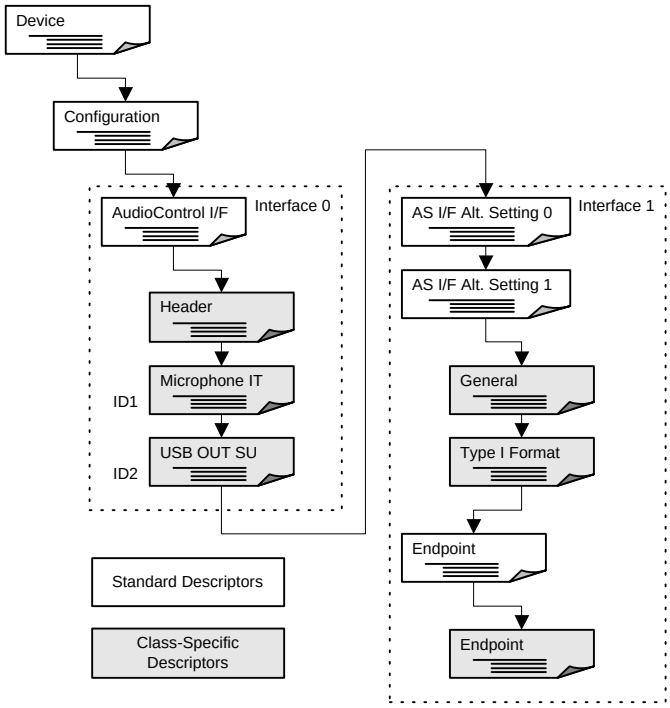

Figure B-2: USB Microphone Descriptor Hierarchy

B.3 Descriptors

The following sections present all the descriptors that are used to describe the device.

B.3.1 Device Descriptor

Table B-1: USB Microphone Device Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	0x12	Size of this descriptor, in bytes.
1	bDescriptorType	1	0x01	DEVICE descriptor.
2	bcdUSB	2	0x0100	1.00 - current revision of USB specification.
4	bDeviceClass	1	0x00	Device defined at Interface level.
5	bDeviceSubClass	1	0x00	Unused.
6	bDeviceProtocol	1	0x00	Unused.
7	bMaxPacketSize0	1	0x08	8 bytes.
8	idVendor	2	0xXXXX	Vendor ID.
10	idProduct	2	0xXXXX	Product ID.

Offset	Field	Size	Value	Description
12	bcdDevice	2	0xXXXX	Device Release Code.
14	iManufacturer	1	0x01	Index to string descriptor that contains the string <Your Name> in Unicode.
15	iProduct	1	0x02	Index to string descriptor that contains the string <Your Product Name> in Unicode.
16	iSerialNumber	1	0x00	Unused.
17	bNumConfigurations	1	0x01	One configuration.

B.3.2 Configuration Descriptor

Table B-2: USB Microphone Configuration Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	0x09	Size of this descriptor, in bytes.
1	bDescriptorType	1	0x02	CONFIGURATION descriptor.
2	wTotalLength	2	0x0064	Length of the total configuration block, including this descriptor, in bytes.
4	bNumInterfaces	1	0x02	Two interfaces.
5	bConfigurationValue	1	0x01	ID of this configuration.
6	iConfiguration	1	0x00	Unused.
7	bmAttributes	1	0x80	Bus Powered device, not Self Powered, no Remote wakeup capability.
8	MaxPower	1	0x0A	20 mA Max. power consumption.

B.3.3 AudioControl Interface Descriptor

The AudioControl interface describes the device structure (audio function topology) and is used to manipulate the Audio Controls.

B.3.3.1 Standard AC Interface Descriptor

The AudioControl interface has no dedicated endpoints associated with it. It uses the default pipe (endpoint 0) for all communication purposes. Class-specific AudioControl Requests are sent using the default pipe. There is no Status Interrupt endpoint provided.

Table B-3: USB Microphone Standard AC Interface Descriptor

Offset	Field	Size	Value	Description

Offset	Field	Size	Value	Description
0	bLength	1	0x09	Size of this descriptor, in bytes.
1	bDescriptorType	1	0x04	INTERFACE descriptor.
2	bInterfaceNumber	1	0x00	Index of this interface.
3	bAlternateSetting	1	0x00	Index of this setting.
4	bNumEndpoints	1	0x00	0 endpoints.
5	bInterfaceClass	1	0x01	AUDIO.
6	bInterfaceSubclass	1	0x01	AUDIO_CONTROL.
7	bInterfaceProtocol	1	0x00	Unused.
8	iInterface	1	0x00	Unused.

B.3.3.2 Class-specific AC Interface Descriptor

The Class-specific AC interface descriptor is always headed by a Header descriptor that contains general information about the AudioControl interface. It contains all the pointers needed to describe the Audio Interface Collection, associated with the described audio function.

Table B-4: USB Microphone Class-specific AC Interface Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	0x09	Size of this descriptor, in bytes.
1	bDescriptorType	1	0x24	CS_INTERFACE.
2	bDescriptorSubtype	1	0x01	HEADER subtype.
3	bcdADC	2	0x0100	Revision of class specification - 1.0
5	wTotalLength	2	0x001E	Total size of class specific descriptors.
7	bInCollection	1	0x01	Number of streaming interfaces.
8	baInterfaceNr(1)	1	0x01	AudioStreaming interface 1 belongs to this AudioControl interface.

B.3.3.3 Input Terminal Descriptor

This descriptor describes the Input Terminal that represents the microphone capsule, followed by the A-to-D converter. The resulting digital audio stream leaves the Input Terminal through the single Output Pin. The audio channel cluster contains a single logical channel (bNrChannels=1) and there is no spatial location associated with this mono channel (wChannelConfig=0x0000).

Table B-5: USB Microphone Input Terminal Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	0x0C	Size of this descriptor, in bytes.
1	bDescriptorType	1	0x24	CS_INTERFACE.
2	bDescriptorSubtype	1	0x02	INPUT_TERMINAL subtype.
3	bTerminalID	1	0x01	ID of this Input Terminal.
4	wTerminalType	2	0x0201	Terminal is Microphone.
6	bAssocTerminal	1	0x00	No association.
7	bNrChannels	1	0x01	One channel.
8	wChannelConfig	2	0x0000	Mono sets no position bits.
10	iChannelNames	1	0x00	Unused.
11	iTerminal	1	0x00	Unused.

B.3.3.4 Output Terminal Descriptor

This descriptor describes the Output Terminal that represents the USB pipe to the Host PC. Its Input Pin is directly connected to the Output Pin of the Input Terminal (bSourceID= Input Terminal ID).

Table B-6: USB Microphone Output Terminal Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	0x09	Size of this descriptor, in bytes.
1	bDescriptorType	1	0x24	CS_INTERFACE.
2	bDescriptorSubtype	1	0x03	OUTPUT_TERMINAL subtype.
3	bTerminalID	1	0x02	ID of this Output Terminal.
4	wTerminalType	2	0x0101	USB Streaming.
6	bAssocTerminal	1	0x00	Unused.
7	bSourceID	1	0x01	From Input Terminal.
8	iTerminal	1	0x00	Unused.

B.3.4 AudioStreaming Interface Descriptor

The AudioStreaming interface has two possible alternate settings.

B.3.4.1 Zero-bandwidth Alternate Setting 0

Alternate setting 0 is a zero-bandwidth setting, used to relinquish the claimed bandwidth on the bus when the microphone is not in use. It is the default setting after power-up. The zero bandwidth is implemented by specifying that this alternate setting of the interface has no endpoints associated with it (bNumEndpoints=0). The collection of descriptors for this alternate setting reduces to the standard interface descriptor.

B.3.4.1.1.1 Standard AS Interface Descriptor

Table B-7: USB Microphone Standard AS Interface Descriptor (Alt. Set. 0)

Offset	Field	Size	Value	Description
0	bLength	1	0x09	Size of this descriptor, in bytes.
1	bDescriptorType	1	0x04	INTERFACE descriptor.
2	bInterfaceNumber	1	0x01	Index of this interface.
3	bAlternateSetting	1	0x00	Index of this alternate setting.
4	bNumEndpoints	1	0x00	0 endpoints.
5	bInterfaceClass	1	0x01	AUDIO.
6	bInterfaceSubclass	1	0x02	AUDIO_STREAMING.
7	bInterfaceProtocol	1	0x00	Unused.
8	iInterface	1	0x00	Unused.

B.3.4.2 Operational Alternate Setting 1

Alternate setting 1 is the operational setting of the interface. It contains the standard and class-specific interface and endpoint descriptors.

B.3.4.2.1.1 Standard AS Interface Descriptor

Table B-8: USB Microphone Standard AS Interface Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	0x09	Size of this descriptor, in bytes.
1	bDescriptorType	1	0x04	INTERFACE descriptor.
2	bInterfaceNumber	1	0x01	Index of this interface.
3	bAlternateSetting	1	0x01	Index of this alternate setting.
4	bNumEndpoints	1	0x01	One endpoint.
5	bInterfaceClass	1	0x01	AUDIO.

Offset	Field	Size	Value	Description
6	bInterfaceSubclass	1	0x02	AUDIO_STREAMING.
7	bInterfaceProtocol	1	0x00	Unused.
8	iInterface	1	0x00	Unused.

B.3.4.2.1.2 Class-specific AS General Interface Descriptor

Table B-9: USB Microphone Class-specific AS General Interface Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	0x07	Size of this descriptor, in bytes.
1	bDescriptorType	1	0x24	CS_INTERFACE descriptor.
2	bDescriptorSubtype	1	0x01	GENERAL subtype.
3	bTerminalLink	1	0x02	Unit ID of the Output Terminal.
4	bDelay	1	0x01	Interface delay.
5	wFormatTag	2	0x0001	PCM Format.

B.3.4.2.1.3 Type I Format Type Descriptor

Table B-10: USB Microphone Type I Format Type Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	0x0B	Size of this descriptor, in bytes.
1	bDescriptorType	1	0x24	CS_INTERFACE descriptor.
2	bDescriptorSubtype	1	0x02	FORMAT_TYPE subtype.
3	bFormatType	1	0x01	FORMAT_TYPE_I.
4	bNrChannels	1	0x01	One channel.
5	bSubFrameSize	1	0x02	Two bytes per audio subframe.
6	bBitResolution	1	0x10	16 bits per sample.
7	bSamFreqType	1	0x01	One frequency supported.
8	tSamFreq	3	0x01F40	8000Hz.

B.3.4.2.1.4 Standard Endpoint Descriptor

Table B-11: USB Microphone Standard Endpoint Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	0x09	Size of this descriptor, in bytes.
1	bDescriptorType	1	0x05	ENDPOINT descriptor.
2	bEndpointAddress	1	0x81	IN Endpoint 1.
3	bmAttributes	1	0x01	Isochronous, not shared.
4	wMaxPacketSize	2	0x0010	16 bytes per packet.
6	bInterval	1	0x01	One packet per frame.
7	bRefresh	1	0x00	Unused.
8	bSynchAddress	1	0x00	Unused.

B.3.4.2.1.5 Class-specific Isochronous Audio Data Endpoint Descriptor

Table B-12: USB Microphone Class-specific Isoc. Audio Data Endpoint Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	0x07	Size of this descriptor, in bytes.
1	bDescriptorType	1	0x25	CS_ENDPOINT descriptor
2	bDescriptorSubtype	1	0x01	GENERAL subtype.
3	bmAttributes	1	0x00	No sampling frequency control, no pitch control, no packet padding.
4	bLockDelayUnits	1	0x00	Unused.
5	wLockDelay	2	0x0000	Unused.

B.3.5 String Descriptors

There are two string descriptors available. The first string descriptor contains the Manufacturer information and the second one contains Product related information. The following sections present an example of how these descriptors could look like.

B.3.5.1 Manufacturer String Descriptor

Table B-13: USB Microphone Manufacturer String Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	0x18	Size of this descriptor, in bytes.
1	bDescriptorType	1	0x03	STRING descriptor.

Offset	Field	Size	Value	Description
2	bString	1	0x0054 0x0048 0x0045 0x0020 0x0043 0x004F 0x004D 0x0050 0x0041 0x004E 0x0059	“THE COMPANY”

B.3.5.2 Product String Descriptor

Table B-14: USB Microphone Product String Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	0x18	Size of this descriptor, in bytes.
1	bDescriptorType	1	0x03	STRING descriptor.
2	bString	1	0x004D 0x0069 0x0063 0x0072 0x006F 0x0070 0x0068 0x006F 0x006E 0x0065	“Microphone”

B.4 Requests

B.4.1 Standard Requests

The microphone supports all necessary standard requests.

B.4.2 Class-specific Requests

There are no class-specific requests supported.

Appendix C. Example 2: USB Telephone (Informative)

C.1 Product Description

This is a USB Telephone with 16-bit 8KHz input and output. It has a handset and Phone line in and out connectors. Selector Units are used to allow the Host to talk directly to the telephone line or to allow the handset to be used with the Host for an Internet telephone call. This is an analog device. The telephone line and handset signals are analog and all switching is performed in the analog domain. The digital USB audio stream, coming from AudioStreaming interface 1 is converted to the analog domain before entering the audio function through Input Terminal ID3. Likewise, the analog signal coming from Selector Unit ID9 is delivered to Output Terminal ID6 and converted back to the digital domain before it is delivered to AudioStreaming interface 2.

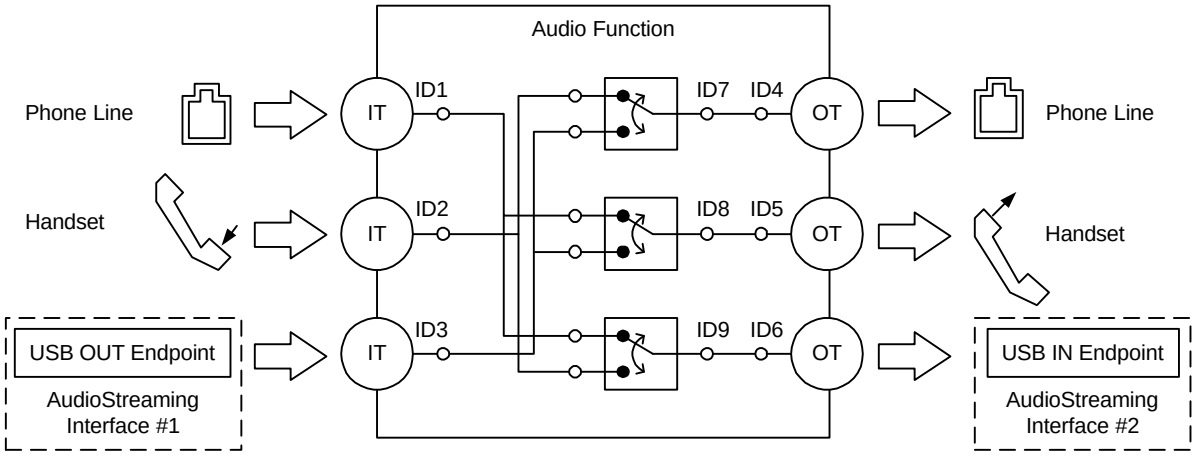

Figure C-1: USB Telephone Topology

It is a neat device, but telephones that are much more elaborate could be constructed with speakerphone and acoustic echo cancellation incorporated.

C.2 Descriptor Hierarchy

This USB Telephone device includes an AudioControl interface (0) and two AudioStreaming interfaces (1&2). The AudioStreaming interfaces both feature two alternate settings. The first alternate setting (Alternate Setting 0) has zero bandwidth associated with it so that switching to this alternate setting effectively frees all allocated bandwidth on the USB for this device. Zero bandwidth is indicated by the lack of a streaming endpoint. Alternate Setting 1 is the operational part of the interface and it has one isochronous endpoint. Figure presents the descriptor hierarchy.

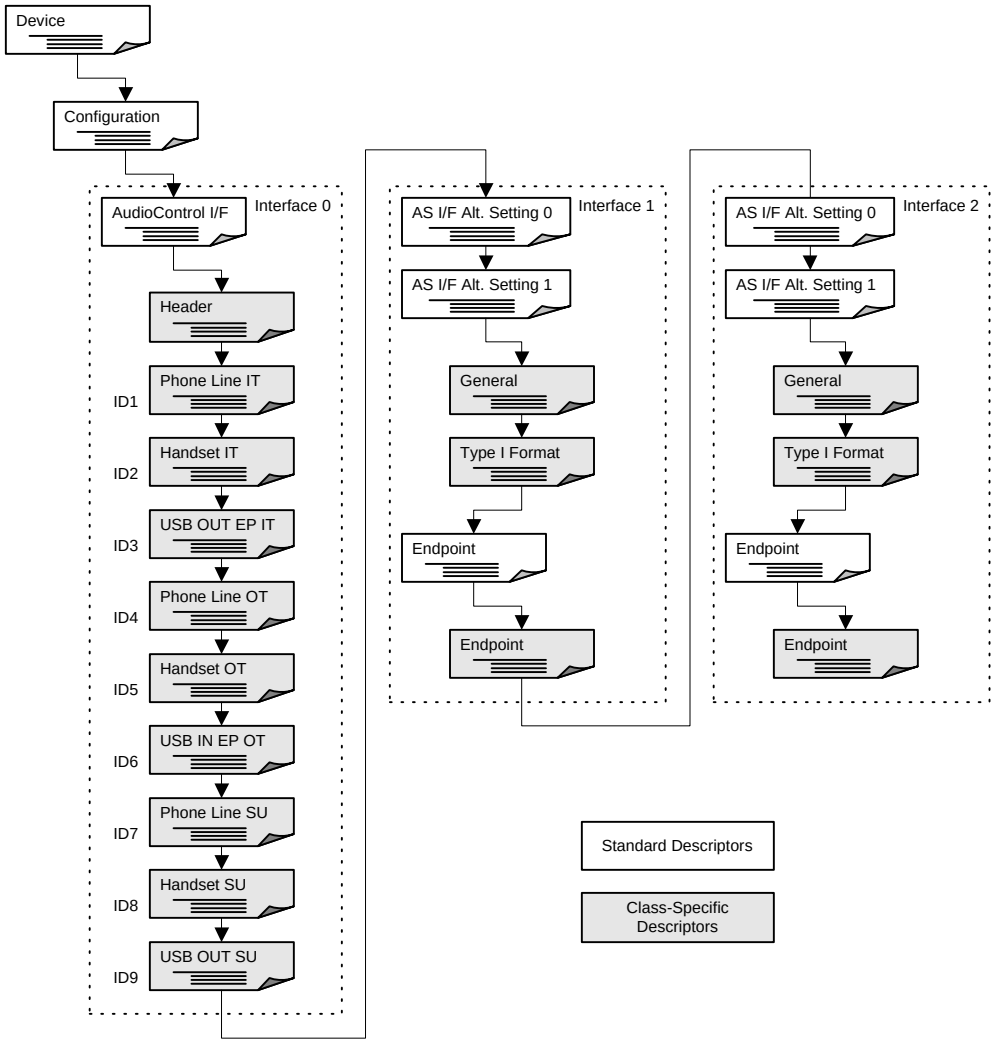

Figure C-2: USB Telephone Descriptor Hierarchy

C.3 Descriptors

The following sections present all the descriptors that are used to describe the device.

C.3.1 Device Descriptor

Table C-1: USB Telephone Device Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	0x12	Size of this descriptor, in bytes.
1	bDescriptorType	1	0x01	DEVICE descriptor.
2	bcdUSB	2	0x0100	1.00 - current revision of USB spec.
4	bDeviceClass	1	0x00	Device defined at Interface level.

Offset	Field	Size	Value	Description
5	bDeviceSubClass	1	0x00	Unused.
6	bDeviceProtocol	1	0x00	Unused.
7	bMaxPacketSize0	1	0x08	8 bytes.
8	idVendor	2	0xXXXX	Vendor ID.
10	idProduct	2	0xXXXX	Product ID.
12	bcdDevice	2	0xXXXX	Device Release Code.
14	iManufacturer	1	0x01	Index to string descriptor that contains the string <Your Name> in Unicode.
15	iProduct	1	0x02	Index to string descriptor that contains the string <Your Product Name> in Unicode.
16	iSerialNumber	1	0x00	Unused.
17	bNumConfigurations	1	0x01	One configuration.

C.3.2 Configuration Descriptor

Table C-2: USB Telephone Configuration Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	0x09	Size of this descriptor, in bytes.
1	bDescriptorType	1	0x02	CONFIGURATION descriptor.
2	wTotalLength	2	0x00XX	Length of the total configuration block, including this descriptor, in bytes.
4	bNumInterfaces	1	0x03	Three interfaces
5	bConfigurationValue	1	0x01	ID of this configuration
6	iConfiguration	1	0x00	Unused.
7	bmAttributes	1	0x60	Self Powered Remote Wakeup capable.
8	MaxPower	1	0x00	Not applicable.

C.3.3 AudioControl Interface Descriptor

The AudioControl interface describes the device structure and is used to manipulate the Audio Controls.

C.3.3.1 Standard AC Interface Descriptor

The AudioControl interface has no dedicated endpoints associated with it. It uses the default pipe (endpoint 0) for all communication purposes. Class-specific AudioControl Requests are sent using the default pipe. There is no Status Interrupt endpoint provided.

Table C-3: USB Telephone Standard AC Interface Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	0x09	Size of this descriptor, in bytes.
1	bDescriptorType	1	0x04	INTERFACE descriptor.
2	bInterfaceNumber	1	0x00	Index of this interface.
3	bAlternateSetting	1	0x00	Index of this setting.
4	bNumEndpoints	1	0x00	0 endpoints.
5	bInterfaceClass	1	0x01	AUDIO.
6	bInterfaceSubclass	1	0x01	AUDIO_CONTROL.
7	bInterfaceProtocol	1	0x00	Unused.
8	iInterface	1	0x00	Unused.

C.3.3.2 Class-specific Interface Descriptor

The Class-specific AC interface descriptor is always headed by a Header descriptor that contains general information about the AudioControl interface. It contains all the pointers needed to describe the Audio Interface Collection, associated with the described audio function.

Table C-4: USB Telephone Class-specific Interface Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	0x0A	Size of this descriptor, in bytes.
1	bDescriptorType	1	0x24	CS_INTERFACE.
2	bDescriptorSubtype	1	0x01	HEADER subtype.
3	bcdADC	2	0x0100	Revision of class specification - 1.0
5	wTotalLength	2	0x0064	Total size of class specific descriptors.
7	bInCollection	1	0x02	Number of streaming interfaces
8	baInterfaceNr(1)	1	0x01	AudioStreaming interface 1 belongs to this AudioControl interface.
9	BaInterfaceNr(2)	1	0x02	AudioStreaming interface 2 belongs to this AudioControl interface.

C.3.3.3 Input Terminal Descriptor (ID1)

This descriptor describes the Input Terminal that represents the analog telephone line input. The audio channel cluster on the single Output Pin contains a single logical channel (bNrChannels=1) and there is no spatial location associated with this mono channel (wChannelConfig=0x0000).

This is the input part of a bi-directional Terminal and therefore has an associated Output Terminal (ID4).

Table C-5: USB Telephone Input Terminal Descriptor (ID1)

Offset	Field	Size	Value	Description
0	bLength	1	0x0C	Size of this descriptor, in bytes.
1	bDescriptorType	1	0x24	CS_INTERFACE.
2	bDescriptorSubtype	1	0x02	INPUT_TERMINAL subtype.
3	bTerminalID	1	0x01	ID of this Terminal.
4	wTerminalType	2	0x0501	Terminal is Phone Line In.
6	bAssocTerminal	1	0x04	Associated with Phone Line Out Terminal.
7	bNrChannels	1	0x01	One channel.
8	wChannelConfig	2	0x0000	Mono sets no position bits.
10	iChannelNames	1	0x00	Unused.
11	iTerminal	1	0x00	Unused.

C.3.3.4 Input Terminal Descriptor (ID2)

This descriptor describes the telephone handset input microphone. The audio channel cluster on the single Output Pin contains a single logical channel (bNrChannels=1) and there is no spatial location associated with this mono channel (wChannelConfig=0x0000).

This is the input part of a bi-directional Terminal and therefore has an associated Output Terminal (ID5).

Table C-6: USB Telephone Input Terminal Descriptor (ID2)

Offset	Field	Size	Value	Description
0	bLength	1	0x0C	Size of this descriptor, in bytes.
1	bDescriptorType	1	0x24	CS_INTERFACE.
2	bDescriptorSubtype	1	0x02	INPUT_TERMINAL subtype.
3	bTerminalID	1	0x02	ID of this Terminal.
4	wTerminalType	2	0x0401	Terminal is Handset In.
6	bAssocTerminal	1	0x05	Associated with Handset Out Terminal.

Offset	Field	Size	Value	Description
7	bNrChannels	1	0x01	One channel.
8	wChannelConfig	2	0x0000	Mono sets no position bits.
10	iChannelNames	1	0x00	Unused.
11	iTerminal	1	0x04	Unused.

C.3.3.5 Input Terminal Descriptor (ID3)

This descriptor describes the USB stream from the Host to the telephone set. The audio channel cluster on the single Output Pin contains a single logical channel (bNrChannels=1) and there is no spatial location associated with this mono channel (wChannelConfig=0x0000).

This is the input part of a bi-directional Terminal and therefore has an associated Output Terminal (ID6).

Table C-7: USB Telephone Input Terminal Descriptor (ID3)

Offset	Field	Size	Value	Description
0	bLength	1	0x0C	Size of this descriptor, in bytes.
1	bDescriptorType	1	0x24	CS_INTERFACE.
2	bDescriptorSubtype	1	0x02	INPUT_TERMINAL subtype.
3	bTerminalID	1	0x03	ID of this Terminal.
4	wTerminalType	2	0x0101	Terminal is USB Streaming In.
6	bAssocTerminal	1	0x06	Associated with USB Streaming out Terminal.
7	bNrChannels	1	0x01	One channel.
8	wChannelConfig	2	0x0000	Mono sets no position bits.
10	iChannelNames	1	0x00	Unused.
11	iTerminal	1	0x05	Unused.

C.3.3.6 Output Terminal Descriptor (ID4)

This descriptor describes the Output Terminal that represents the analog telephone line output. The audio channel cluster on the single Input Pin contains a single logical channel.

This is the output part of a bi-directional Terminal and therefore has an associated Input Terminal (ID1).

Table C-8: USB Telephone Output Terminal Descriptor (ID4)

Offset	Field	Size	Value	Description
0	bLength	1	0x09	Size of this descriptor, in bytes.

Offset	Field	Size	Value	Description
1	bDescriptorType	1	0x24	CS_INTERFACE.
2	bDescriptorSubtype	1	0x03	OUTPUT_TERMINAL subtype.
3	bTerminalID	1	0x04	ID of this Terminal.
4	wTerminalType	2	0x0501	Terminal is Phone Line Out.
6	bAssocTerminal	1	0x01	Associated with Phone Line In Terminal.
7	bSourceID	1	0x07	From Phone Line Selector Unit.
8	iTerminal	1	0x06	Unused.

C.3.3.7 Output Terminal Descriptor (ID5)

This descriptor describes the telephone handset output earpiece. The audio channel cluster on the single Input Pin contains a single logical channel.

This is the output part of a bi-directional Terminal and therefore has an associated Input Terminal (ID2).

Table C-9: USB Telephone Output Terminal Descriptor (ID5)

Offset	Field	Size	Value	Description
0	bLength	1	0x09	Size of this descriptor, in bytes.
1	bDescriptorType	1	0x24	CS_INTERFACE.
2	bDescriptorSubtype	1	0x03	OUTPUT_TERMINAL subtype.
3	bTerminalID	1	0x05	ID of this Terminal.
4	wTerminalType	2	0x0401	Terminal is Handset Out.
6	bAssocTerminal	1	0x01	Associated with Handset In Terminal.
7	bSourceID	1	0x08	From Handset Selector Unit.
8	iTerminal	1	0x00	Unused.

C.3.3.8 Output Terminal Descriptor (ID6)

This descriptor describes the USB stream from the telephone set to the Host. The audio channel cluster on the single Input Pin contains a single logical channel.

This is the output part of a bi-directional Terminal and therefore has an associated Input Terminal (ID3).

Table C-10: USB Telephone Output Terminal Descriptor (ID6)

Offset	Field	Size	Value	Description
0	bLength	1	0x09	Size of this descriptor, in bytes.

Offset	Field	Size	Value	Description
1	bDescriptorType	1	0x24	CS_INTERFACE.
2	bDescriptorSubtype	1	0x03	OUTPUT_TERMINAL subtype.
3	bTerminalID	1	0x06	ID of this Terminal.
4	wTerminalType	2	0x0101	Terminal is USB Streaming Out.
6	bAssocTerminal	1	0x03	Associated with USB Streaming In Terminal.
7	bSourceID	1	0x09	From USB Selector Unit.
8	iTerminal	1	0x00	Unused.

C.3.3.9 Selector Unit Descriptor (ID7)

This descriptor describes the Selector Unit connected to the Phone Line Out Output Terminal. Either Handset In or USB In signals can be selected.

Table C-11: USB Telephone Selector Unit Descriptor (ID7)

Offset	Field	Size	Value	Description
0	bLength	1	0x09	Size of this descriptor, in bytes.
1	bDescriptorType	1	0x24	CS_INTERFACE.
2	bDescriptorSubtype	1	0x05	SELECTOR_UNIT subtype.
3	bUnitD	1	0x07	ID of this Unit.
4	bNrInPins	1	0x02	Number of input pins.
6	baSourceID(1)	1	0x02	From Handset In Terminal.
7	baSourceID(2)	1	0x03	From USB Streaming In Terminal.
8	iSelector	1	0x00	Unused.

C.3.3.10 Selector Unit Descriptor (ID8)

This descriptor describes the Selector Unit connected to the Handset Out Output Terminal. Either Phone Line In or USB In signals can be selected.

Table C-12: USB Telephone Selector Unit Descriptor (ID8)

Offset	Field	Size	Value	Description
0	bLength	1	0x09	Size of this descriptor, in bytes.
1	bDescriptorType	1	0x24	CS_INTERFACE.

Offset	Field	Size	Value	Description
2	bDescriptorSubtype	1	0x05	SELECTOR_UNIT subtype.
3	bUnitD	1	0x08	ID of this Unit.
4	bNrInPins	1	0x02	Number of input pins.
6	baSourceID(1)	1	0x01	From Phone Line In Terminal.
7	baSourceID(2)	1	0x03	From USB Streaming In Terminal.
8	iSelector	1	0x00	Unused.

C.3.3.11 Selector Unit Descriptor (ID9)

This descriptor describes the Selector Unit connected to the USB Streaming Out Output Terminal. Either Phone Line In or Handset In signals can be selected.

Table C-13: USB Telephone Selector Unit Descriptor (ID9)

Offset	Field	Size	Value	Description
0	bLength	1	0x09	Size of this descriptor, in bytes.
1	bDescriptorType	1	0x24	CS_INTERFACE.
2	bDescriptorSubtype	1	0x05	SELECTOR_UNIT subtype.
3	bUnitD	1	0x09	ID of this Unit
4	bNrInPins	1	0x02	Number of input pins.
6	baSourceID(1)	1	0x01	From Phone Line In Terminal.
7	baSourceID(2)	1	0x02	From Handset In Terminal.
8	iSelector	1	0x00	Unused.

C.3.4 AudioStreaming Interface 1 Descriptor

The AudioStreaming interface 1 is used for streaming audio from the Host to the USB Telephone device. AudioStreaming interface 1 has two alternate settings. The first is the zero bandwidth alternate setting (Alternate Setting 0), used to reclaim USB bandwidth. It is also the default alternate setting. The second alternate setting (1) is the fully operational setting for the Host to Device streaming communication.

C.3.4.1 Zero-bandwidth Alternate Setting 0

C.3.4.1.1.1 Standard Interface Descriptor

Table C-14: USB Telephone Standard Interface Descriptor (Alt. Set. 0)

Offset	Field	Size	Value	Description
0	bLength	1	0x09	Size of this descriptor, in bytes.
1	bDescriptorType	1	0x04	INTERFACE descriptor.
2	bInterfaceNumber	1	0x01	Index of this interface.
3	bAlternateSetting	1	0x00	Index of this setting.
4	bNumEndpoints	1	0x00	0 endpoints.
5	bInterfaceClass	1	0x01	AUDIO.
6	bInterfaceSubclass	1	0x02	AUDIO_STREAMING.
7	bInterfaceProtocol	1	0x00	Unused.
8	iInterface	1	0x00	Unused.

C.3.4.2 Operational Alternate Setting 1

C.3.4.2.1.1 Standard AS Interface Descriptor

Table C-15: USB Telephone Standard AS Interface Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	0x09	Size of this descriptor, in bytes.
1	bDescriptorType	1	0x04	INTERFACE descriptor.
2	bInterfaceNumber	1	0x01	Index of this interface.
3	bAlternateSetting	1	0x01	Index of this setting.
4	bNumEndpoints	1	0x01	1 endpoint.
5	bInterfaceClass	1	0x01	AUDIO.
6	bInterfaceSubclass	1	0x02	AUDIO_STREAMING.
7	bInterfaceProtocol	1	0x00	Unused.
8	iInterface	1	0x00	Unused.

C.3.4.2.1.2 Class-specific AS Interface Descriptor

Table C-16: USB Telephone Class-specific AS Interface Descriptor

Offset	Field	Size	Value	Description

Offset	Field	Size	Value	Description
0	bLength	1	0x07	Size of this descriptor, in bytes.
1	bDescriptorType	1	0x24	CS_INTERFACE descriptor.
2	bDescriptorSubtype	1	0x01	GENERAL.
3	bTerminalLink	1	0x03	Linked to USB Streaming In Terminal.
4	bDelay	1	0x01	Interface delay.
5	wFormatTag	2	0x0001	PCM format.

C.3.4.2.1.3 Type I Format Type Descriptor

Table C-17: USB Telephone Type I Format Type Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	0x0B	Size of this descriptor, in bytes.
1	bDescriptorType	1	0x24	CS_INTERFACE.
2	bDescriptorSubtype	1	0x02	FORMAT_TYPE.
3	bFormatType	1	0x01	FORMAT_TYPE_I.
4	bNrChannels	1	0x01	One channel.
5	bSubFrameSize	1	0x02	Two bytes per slot.
6	bBitResolution	1	0x10	16 bits.
7	bSamFreqType	1	0x01	One sampling frequency.
8	tSamFreq	3	0x01F40	8000Hz is the sampling frequency.

C.3.4.2.1.4 Standard Endpoint Descriptor

Table C-18: USB Telephone Standard Endpoint Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	0x09	Size of this descriptor, in bytes.
1	bDescriptorType	1	0x05	ENDPOINT descriptor.
2	bEndpointAddress	1	0x01	OUT Endpoint 1.
3	bmAttributes	1	0x0B	Isochronous transfer type, synchronous synchronization type.
4	wMaxPacketSize	2	0x0010	16 bytes per packet (8 two-byte

Offset	Field	Size	Value	Description
				samples).
6	bInterval	1	0x01	One packet every frame.
7	bRefresh	1	0x00	Unused.
8	bSynchAddress	1	0x00	Unused.

C.3.4.2.1.5 Class-specific Isochronous Audio Data Endpoint Descriptor

Table C-19: USB Telephone Class-specific Isoc. Audio Data Endpoint Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	0x07	Size of this descriptor, in bytes.
1	bDescriptorType	1	0x25	CS_ENDPOINT.
2	bDescriptorSubtype	1	0x01	GENERAL.
3	bmAttributes	1	0x00	No sampling frequency control, no pitch control.
4	bLockDelayUnits	1	0x00	Unused.
5	wLockDelay	2	0x0000	Unused.

C.3.5 AudioStreaming Interface 2 Descriptor

The AudioStreaming interface 2 is used for streaming audio from the USB Telephone device to the Host. AudioStreaming interface 2 has two alternate settings. The first is the zero bandwidth alternate setting (Alternate Setting 0), used to reclaim USB bandwidth. It is also the default alternate setting. The second alternate setting (1) is the fully operational setting for the Device to Host streaming communication.

C.3.5.1 Zero-bandwidth Alternate Setting 0

C.3.5.1.1.1 Standard Interface Descriptor

Table C-20: USB Telephone Standard Interface Descriptor (Alt. Set. 0)

Offset	Field	Size	Value	Description
0	bLength	1	0x09	Size of this descriptor, in bytes.
1	bDescriptorType	1	0x04	INTERFACE descriptor.
2	bInterfaceNumber	1	0x01	Index of this interface.
3	bAlternateSetting	1	0x00	Index of this setting.
4	bNumEndpoints	1	0x00	0 endpoints.

Offset	Field	Size	Value	Description
5	bInterfaceClass	1	0x01	AUDIO.
6	bInterfaceSubclass	1	0x02	AUDIO_STREAMING.
7	bInterfaceProtocol	1	0x00	Unused.
8	iInterface	1	0x00	Unused.

C.3.5.2 Operational Alternate Setting 1

C.3.5.2.1.1 Standard AS Interface Descriptor

Table C-21: USB Telephone Standard AS Interface Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	0x09	Size of this descriptor, in bytes.
1	bDescriptorType	1	0x04	INTERFACE descriptor.
2	bInterfaceNumber	1	0x01	Index of this interface.
3	bAlternateSetting	1	0x01	Index of this setting.
4	bNumEndpoints	1	0x01	1 endpoint.
5	bInterfaceClass	1	0x01	AUDIO.
6	bInterfaceSubclass	1	0x02	AUDIO_STREAMING.
7	bInterfaceProtocol	1	0x00	Unused.
8	iInterface	1	0x00	Unused.

C.3.5.2.1.2 Class-specific AS Interface Descriptor

Table C-22: USB Telephone Class-specific AS Interface Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	0x07	Size of this descriptor, in bytes.
1	bDescriptorType	1	0x24	CS_INTERFACE descriptor.
2	bDescriptorSubtype	1	0x01	GENERAL.
3	bTerminalLink	1	0x06	USB Streaming Out Terminal
4	bDelay	1	0x01	Interface delay.
5	wFormatTag	2	0x0001	PCM format.

C.3.5.2.1.3 Type I format type descriptor

Table C-23: USB Telephone Type I format type descriptor

Offset	Field	Size	Value	Description
0	bLength	1	0x0B	Size of this descriptor, in bytes.
1	bDescriptorType	1	0x24	CS_INTERFACE.
2	bDescriptorSubtype	1	0x02	FORMAT_TYPE.
3	bFormatType	1	0x01	FORMAT_TYPE_I.
4	bNrChannels	1	0x01	One channel.
5	bSubFrameSize	1	0x02	Two bytes per slot.
6	bBitResolution	1	0x10	16 bits.
7	bSamFreqType	1	0x01	One sampling frequency.
8	tSamFreq	3	0x01F40	8000Hz is the sampling frequency.

C.3.5.2.1.4 Standard Endpoint descriptor

Table C-24: USB Telephone Standard Endpoint descriptor

Offset	Field	Size	Value	Description
0	bLength	1	0x09	Size of this descriptor, in bytes.
1	bDescriptorType	1	0x05	ENDPOINT descriptor.
2	bEndpointAddress	1	0x81	IN Endpoint 1.
3	bmAttributes	1	0x0B	Isochronous transfer type, synchronous synchronization type.
4	wMaxPacketSize	2	0x0010	16 bytes (8 two-byte samples).
6	bInterval	1	0x01	One packet every frame.
7	bRefresh	1	0x00	Unused
8	bSynchAddress	1	0x00	Unused.

C.3.5.2.1.5 Class-specific Isochronous Audio Data Endpoint Descriptor

Table C-25: USB Telephone Class-specific Isoc. Audio Data Endpoint Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	0x07	Size of this descriptor, in bytes.

Offset	Field	Size	Value	Description
1	bDescriptorType	1	0x25	CS_ENDPOINT.
2	bDescriptorSubtype	1	0x01	GENERAL.
3	bmAttributes	1	0x00	No sampling frequency control, no pitch control.
4	bLockDelayUnits	1	0x00	Unused.
5	wLockDelay	2	0x0000	Unused.

C.3.6 String Descriptors

There are two string descriptors available. The first string descriptor contains the Manufacturer information and the second one contains Product related information. The following sections present an example of how these descriptors could look like.

C.3.6.1 Manufacturer String Descriptor

Table C-26: USB Telephone Manufacturer String Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	0x18	Size of this descriptor, in bytes.
1	bDescriptorType	1	0x03	STRING descriptor.
2	bString	1	0x0054 0x0048 0x0045 0x0020 0x0043 0x004F 0x004D 0x0050 0x0041 0x004E 0x0059	“THE COMPANY”

C.3.6.2 Product String Descriptor

Table C-27: USB Telephone Product String Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	0x16	Size of this descriptor, in bytes.
1	bDescriptorType	1	0x03	STRING descriptor.
2	bString	1	0x0054 0x0065 0x006C 0x0065	“Telephone”

Offset	Field	Size	Value	Description
			0x0070 0x0068 0x006F 0x006E 0x0065	

C.4 Requests

C.4.1 Standard requests

All standard Requests, necessary to operate the device are supported. The next section presents the Set Interface Request as an example.

C.4.1.1 Set interface

This request selects the alternate setting on interface 1 or 2 to control bandwidth allocation.

Table 5-28: Set Interface Request Values

Offset	Field	Size	Value	Description
0	bmRequestType	1	0x01	D7: 0 = Host to Device. D6..5: 00 = Standard Request. D4..0: 00001 = Recipient is interface.
1	bRequest	1	0x0B	SET_INTERFACE.
2	wValue	2	0x0000 or 0x0001	0x00 is zero bandwidth alternate setting. 0x01 is normal isochronous operation.
4	wIndex	2	0x0001 or 0x0002	Interface number of one of the AudioStreaming interfaces.
6	wLength	2	0x0000	No Parameter Block

C.4.2 Class-specific Requests

The only class-specific Request supported is the Set/Get Selector Control Request. The following sections describe these requests in detail.

C.4.2.1 Set Selector Unit Control Request

This Request sets the Selector Unit Control to the desired value.

Table C-29: Set Selector Unit Control Request Values

Offset	Field	Size	Value	Description
0	bmRequestType	1	0x21	D7: 0 = Host to Device. D6..5: 01 = Class Request.

Offset	Field	Size	Value	Description
				D4..0: 00001 = Recipient is interface.
1	bRequest	1	0x01	SET_CUR.
2	wValue	2	0x0000	Must be zero.
4	wIndex	2	0x0000	Interface number of the AudioControl interface.
6	wLength	2	0x0001	Parameter Block Length

The one-byte Parameter Block contains the new **bSelector** value for the Selector Control. Since all the Selector Units have two Input Pins, the valid range for **bSelector** is [1,2].

C.4.2.2 Get Selector Unit Control Request

This Request retrieves the Selector Unit Control parameter.

Table C-30: Get Selector Unit Control Request Values

Offset	Field	Size	Value	Description
0	bmRequestType	1	0xA1	D7: 1 = Device to Host. D6..5: 01 = Class Request. D4..0: 00001 = Recipient is interface.
1	bRequest	1	0x81 0x82 0x83 0x84	GET_CUR. GET_MIN. GET_MAX. GET_RES.
2	wValue	2	0x0000	Must be zero.
4	wIndex	2	0x0000	Interface number of the AudioControl interface.
6	wLength	2	0x0001	Parameter Block Length

The actual setting of the Selector Control is returned in the one-byte Parameter Block. Since all the Selector Units have two Input Pins, the valid range for the returned value is [1,2].