

INTERNATIONAL STANDARD

NORME INTERNATIONALE



**Universal serial bus interfaces for data and power –
Part 1-5: Common components – USB Audio 3.0 device class definition**

**Interfaces de bus universel en série pour les données et l'alimentation
électrique –
Partie 1-5: Composants communs – Définition de classes de dispositifs USB
Audio 3.0**

IECNORM.COM : Click to view the full PDF of IEC 62680-1-5:2019



THIS PUBLICATION IS COPYRIGHT PROTECTED
Copyright © 1997-2016 USB-IF

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from IEC, or USB-IF at the respective address given below. Any questions about USB-IF copyright should be addressed to the USB-IF. Enquiries about obtaining additional rights to this publication and other information requests should be addressed to the IEC or your local IEC member National Committee.

IEC Central Office
3, rue de Varembe
CH-1211 Geneva 20
Switzerland
Tel.: +41 22 919 02 11
info@iec.ch
www.iec.ch

USB Implementers Forum, Inc.
3855 S.W. 153rd Drive
Beaverton, OR 97003
United States of America
Tel: +1 503-619-0426
admin@usb.org
www.usb.org

About the IEC

The International Electrotechnical Commission (IEC) is the leading global organization that prepares and publishes International Standards for all electrical, electronic and related technologies.

About IEC publications

The technical content of IEC publications is kept under constant review by the IEC. Please make sure that you have the latest edition, a corrigendum or an amendment might have been published.

IEC publications search - webstore.iec.ch/advsearchform

The advanced search enables to find IEC publications by a variety of criteria (reference number, text, technical committee,...). It also gives information on projects, replaced and withdrawn publications.

IEC Just Published - webstore.iec.ch/justpublished

Stay up to date on all new IEC publications. Just Published details all new publications released. Available online and once a month by email.

IEC Customer Service Centre - webstore.iec.ch/csc

If you wish to give us your feedback on this publication or need further assistance, please contact the Customer Service Centre: sales@iec.ch.

Electropedia - www.electropedia.org

The world's leading online dictionary on electrotechnology, containing more than 22 000 terminological entries in English and French, with equivalent terms in 16 additional languages. Also known as the International Electrotechnical Vocabulary (IEV) online.

IEC Glossary - std.iec.ch/glossary

67 000 electrotechnical terminology entries in English and French extracted from the Terms and Definitions clause of IEC publications issued since 2002. Some entries have been collected from earlier publications of IEC TC 37, 77, 86 and CISPR.

IECNORM.COM : Click to view the full PDF © IEC 62689-1-5:2019

INTERNATIONAL STANDARD

NORME INTERNATIONALE



**Universal serial bus interfaces for data and power –
Part 1-5: Common components – USB Audio 3.0 device class definition**

**Interfaces de bus universel en série pour les données et l'alimentation
électrique –
Partie 1-5: Composants communs – Définition de classes de dispositifs USB
Audio 3.0**

INTERNATIONAL
ELECTROTECHNICAL
COMMISSION

COMMISSION
ELECTROTECHNIQUE
INTERNATIONALE

ICS 33.120.20; 35.200; 29.200

ISBN 978-2-8322-7241-1

**Warning! Make sure that you obtained this publication from an authorized distributor.
Attention! Veuillez vous assurer que vous avez obtenu cette publication via un distributeur agréé.**

INTERNATIONAL ELECTROTECHNICAL COMMISSION

UNIVERSAL SERIAL BUS INTERFACES FOR DATA AND POWER –

Part 1-5: Common components – USB Audio 3.0 device class definition

FOREWORD

- 1) The International Electrotechnical Commission (IEC) is a worldwide organization for standardization comprising all national electrotechnical committees (IEC National Committees). The object of IEC is to promote international co-operation on all questions concerning standardization in the electrical and electronic fields. To this end and in addition to other activities, IEC publishes International Standards, Technical Specifications, Technical Reports, Publicly Available Specifications (PAS) and Guides (hereafter referred to as “IEC Publication(s)”). Their preparation is entrusted to technical committees; any IEC National Committee interested in the subject dealt with may participate in this preparatory work. International, governmental and non-governmental organizations liaising with the IEC also participate in this preparation. IEC collaborates closely with the International Organization for Standardization (ISO) in accordance with conditions determined by agreement between the two organizations.
- 2) The formal decisions or agreements of IEC on technical matters express, as nearly as possible, an international consensus of opinion on the relevant subjects since each technical committee has representation from all interested IEC National Committees.
- 3) IEC Publications have the form of recommendations for international use and are accepted by IEC National Committees in that sense. While all reasonable efforts are made to ensure that the technical content of IEC Publications is accurate, IEC cannot be held responsible for the way in which they are used or for any misinterpretation by any end user.
- 4) In order to promote international uniformity, IEC National Committees undertake to apply IEC Publications transparently to the maximum extent possible in their national and regional publications. Any divergence between any IEC Publication and the corresponding national or regional publication shall be clearly indicated in the latter.
- 5) IEC itself does not provide any attestation of conformity. Independent certification bodies provide conformity assessment services and, in some areas, access to IEC marks of conformity. IEC is not responsible for any services carried out by independent certification bodies.
- 6) All users should ensure that they have the latest edition of this publication.
- 7) No liability shall attach to IEC or its directors, employees, servants or agents including individual experts and members of its technical committees and IEC National Committees for any personal injury, property damage or other damage of any nature whatsoever, whether direct or indirect, or for costs (including legal fees) and expenses arising out of the publication, use of, or reliance upon, this IEC Publication or any other IEC Publications.
- 8) Attention is drawn to the Normative references cited in this publication. Use of the referenced publications is indispensable for the correct application of this publication.
- 9) Attention is drawn to the possibility that some of the elements of this IEC Publication may be the subject of patent rights. IEC shall not be held responsible for identifying any or all such patent rights.

International Standard IEC 62680-1-5 has been prepared by technical area 18: Multimedia home systems and applications for end-user networks, of IEC technical committee 100: Audio, video and multimedia systems and equipment.

The text of this standard was prepared by the USB Implementers Forum (USB-IF). The structure and editorial rules used in this publication reflect the practice of the organization which submitted it.

The text of this International Standard is based on the following documents:

CDV	Report on voting
100/3157/CDV	100/3227/RVC

Full information on the voting for the approval of this International Standard can be found in the report on voting indicated in the above table.

The committee has decided that the contents of this document will remain unchanged until the stability date indicated on the IEC website under "<http://webstore.iec.ch>" in the data related to the specific document. At this date, the document will be

- reconfirmed,
- withdrawn,
- replaced by a revised edition, or
- amended.

IMPORTANT – The 'colour inside' logo on the cover page of this publication indicates that it contains colours which are considered to be useful for the correct understanding of its contents. Users should therefore print this document using a colour printer.

IECNORM.COM : Click to view the full PDF of IEC 62680-1-5:2019

INTRODUCTION

The IEC 62680 series is based on a series of specifications that were originally developed by the USB Implementers Forum (USB-IF). These specifications were submitted to the IEC under the auspices of a special agreement between the IEC and the USB-IF.

This standard is the USB-IF publication USB Device Class Definition for Audio Devices Release 3.0.

The USB Implementers Forum, Inc.(USB-IF) is a non-profit corporation founded by the group of companies that developed the Universal Serial Bus specification. The USB-IF was formed to provide a support organization and forum for the advancement and adoption of Universal Serial Bus technology. The Forum facilitates the development of high-quality compatible USB peripherals (devices), and promotes the benefits of USB and the quality of products that have passed compliance testing.

ANY USB SPECIFICATIONS ARE PROVIDED TO YOU "AS IS", WITH NO WARRANTIES WHATSOEVER, INCLUDING ANY WARRANTY OF MERCHANTABILITY, NON-INFRINGEMENT, OR FITNESS FOR ANY PARTICULAR PURPOSE. THE USB IMPLEMENTERS FORUM AND THE AUTHORS OF ANY USB SPECIFICATIONS DISCLAIM ALL LIABILITY, INCLUDING LIABILITY FOR INFRINGEMENT OF ANY PROPRIETARY RIGHTS, RELATING TO USE OR IMPLEMENTATION OR INFORMATION IN THIS SPECIFICATION.

THE PROVISION OF ANY USB SPECIFICATIONS TO YOU DOES NOT PROVIDE YOU WITH ANY LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS.

Entering into USB Adopters Agreements may, however, allow a signing company to participate in a reciprocal, RAND-Z licensing arrangement for compliant products. For more information, please see:

<https://www.usb.org/documents>

IEC DOES NOT TAKE ANY POSITION AS TO WHETHER IT IS ADVISABLE FOR YOU TO ENTER INTO ANY USB ADOPTERS AGREEMENTS OR TO PARTICIPATE IN THE USB IMPLEMENTERS FORUM.

UNIVERSAL SERIAL BUS DEVICE CLASS DEFINITION FOR AUDIO DEVICES

IECNORM.COM : Click to view the full PDF of IEC 62680-1-5:2019

Release 3.0

September 22, 2016

SCOPE OF THIS RELEASE

This document is the Release 3.0 of this Device Class Definition.

CONTRIBUTORS

Joe Scanlon	Advanced Micro Devices
Rhoads Hollowell	Apple Inc.
Girault Jones	Apple Inc.
Matthew X. Mora	Apple Inc.
Tzung-Dar Tsai	C-Media Electronics, Inc.
Brad Lambert	Cirrus Logic, Inc.
Dan Bogard	Conexant Systems, Inc.
Pete Burgers	DisplayLink (UK), Ltd.
David Roh	Dolby Laboratories, Inc.
Leng Ooi	Google, Inc.
Pierre-Louis Bossart	Intel Corporation
David Hines	Intel Corporation
Abdul Rahman Ismail (Co-Chair)	Intel Corporation
Devon Worrell	Intel Corporation
Chandrashekhar Rao	Logitech, Inc.
Terry Moore	MCCI Corporation
Alex Lin	MediaTek, Inc.
Bala Sivakumar	Microsoft Corporation
Geert Knapen (Co-Chair & Editor)	NXP Semiconductors
	PL Mobile Audio
	411 E. Plumeria drive
	San Jose, CA 95134, USA
	E-mail: geert.knapen@nxp.com
James Goel	Qualcomm, Inc.
Andre Schevciw	Qualcomm, Inc.
Jin-Sheng Wang	Qualcomm, Inc.
Morten Christiansen	Synopsys

REVISION HISTORY

Revision	Date	Filename	Description
1.0	Mar. 18, 98	Audio10.pdf	Release 1.0
2.0	May. 31, 06	Audio20 final.pdf	Release 2.0
3.0	Sep. 22, 16	Audio30.pdf	Release 3.0

**Copyright © 1997-2016 USB Implementers Forum, Inc.
All rights reserved.**

INTELLECTUAL PROPERTY DISCLAIMER

A LICENSE IS HEREBY GRANTED TO REPRODUCE THIS SPECIFICATION FOR INTERNAL USE ONLY. NO OTHER LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, IS GRANTED OR INTENDED HEREBY.

USB-IF AND THE AUTHORS OF THIS SPECIFICATION EXPRESSLY DISCLAIM ALL LIABILITY FOR INFRINGEMENT OF INTELLECTUAL PROPERTY RIGHTS RELATING TO IMPLEMENTATION OF INFORMATION IN THIS SPECIFICATION. USB-IF AND THE AUTHORS OF THIS SPECIFICATION ALSO DO NOT WARRANT OR REPRESENT THAT SUCH IMPLEMENTATION(S) WILL NOT INFRINGE THE INTELLECTUAL PROPERTY RIGHTS OF OTHERS.

THIS SPECIFICATION IS PROVIDED “AS IS” AND WITH NO WARRANTIES, EXPRESS OR IMPLIED, STATUTORY OR OTHERWISE. ALL WARRANTIES ARE EXPRESSLY DISCLAIMED. USB-IF, ITS MEMBERS AND THE AUTHORS OF THIS SPECIFICATION PROVIDE NO WARRANTY OF MERCHANTABILITY, NO WARRANTY OF NON-INFRINGEMENT, NO WARRANTY OF FITNESS FOR ANY PARTICULAR PURPOSE, AND NO WARRANTY ARISING OUT OF ANY PROPOSAL, SPECIFICATION, OR SAMPLE.

IN NO EVENT WILL USB-IF, MEMBERS OR THE AUTHORS BE LIABLE TO ANOTHER FOR THE COST OF PROCURING SUBSTITUTE GOODS OR SERVICES, LOST PROFITS, LOSS OF USE, LOSS OF DATA OR ANY INCIDENTAL, CONSEQUENTIAL, INDIRECT, OR SPECIAL DAMAGES, WHETHER UNDER CONTRACT, TORT, WARRANTY, OR OTHERWISE, ARISING IN ANY WAY OUT OF THE USE OF THIS SPECIFICATION, WHETHER OR NOT SUCH PARTY HAD ADVANCE NOTICE OF THE POSSIBILITY OF SUCH DAMAGES.

NOTE: VARIOUS USB-IF MEMBERS PARTICIPATED IN THE DRAFTING OF THIS SPECIFICATION. CERTAIN OF THESE MEMBERS MAY HAVE DECLINED TO ENTER INTO A SPECIFIC AGREEMENT LICENSING INTELLECTUAL PROPERTY RIGHTS THAT MAY BE INFRINGED IN THE IMPLEMENTATION OF THIS SPECIFICATION. PERSONS IMPLEMENT THIS SPECIFICATION AT THEIR OWN RISK.

Dolby™, AC-3™, Pro Logic™ and Dolby Surround™ are trademarks of Dolby Laboratories, Inc.

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

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

TABLE OF CONTENTS

Scope of This Release	6
Contributors	6
Revision History	6
Table of Contents	8
List of Tables	12
List of Figures	14
1 Introduction	16
1.1 Scope	16
1.2 Purpose	16
1.3 Related Documents	16
1.4 Terms and Abbreviations	16
2 Management Overview	19
2.1 Overview of Key Differences between ADC v2.0 and v3.0	19
3 Functional Characteristics	21
3.1 Introduction	21
3.2 Basic Audio Device Definition	23
3.3 Backwards Compatibility	23
3.4 Audio Interface Association (AIA) and Interface Association Descriptor	23
3.4.1 Audio Function Class	24
3.4.2 Audio Function Subclass	24
3.4.3 Audio Function Protocol	24
3.5 Audio Interface Class	24
3.6 Audio Interface Subclass	24
3.7 Audio Interface Protocol	25
3.8 Audio Function Category	25
3.9 Clock Domains	26
3.10 Power Domains	26
3.11 Audio Synchronization Types	26
3.11.1 Asynchronous	26
3.11.2 Synchronous	26
3.11.3 Adaptive	26
3.11.4 Implications of the Different Synchronization Types	26
3.12 Inter Channel Synchronization	28
3.13 Audio Function Topology	28
3.13.1 Cluster	33
3.13.2 Input Terminal	33
3.13.3 Output Terminal	34
3.13.4 Mixer Unit	35
3.13.5 Selector Unit	35

3.13.6	Feature Unit	36
3.13.7	Sampling Rate Converter Unit.....	36
3.13.8	Effect Unit	37
3.13.9	Processing Unit.....	41
3.13.10	Extension Unit	43
3.13.11	Clock Entities	44
3.14	Operational Model	45
3.14.1	AudioControl Interface.....	46
3.14.2	AudioStreaming Interface	47
3.14.3	Clock Model.....	48
3.14.4	Power Domains Model.....	48
3.14.5	Additional Power Considerations and Requirements	50
3.14.6	Binding between Physical Buttons and Audio Controls	50
4	Descriptors	52
4.1	Standard Descriptors.....	52
4.2	Class-Specific Descriptors.....	52
4.2.1	Traditional Class-Specific Descriptors	53
4.2.2	High Capability Class-Specific Descriptors.....	53
4.3	Cluster Descriptor	54
4.3.1	Cluster Descriptor Header.....	55
4.3.2	Cluster Descriptor Block.....	55
4.3.3	Example Cluster descriptor	63
4.3.4	CEA-861.2 Channel Mapping.....	64
4.4	Physical versus Logical Cluster.....	65
4.4.1	Mapping between Physical and Logical Clusters.....	65
4.5	AudioControl Interface Descriptors	67
4.5.1	Standard AC Interface Descriptor.....	67
4.5.2	Class-Specific AC Interface Descriptor.....	68
4.6	AudioControl Endpoint Descriptors	98
4.6.1	AC Control Endpoint Descriptors	98
4.6.2	AC Interrupt Endpoint Descriptors.....	98
4.7	AudioStreaming Interface Descriptors.....	98
4.7.1	Standard AS Interface Descriptor.....	99
4.7.2	Class-Specific AS Interface Descriptor.....	99
4.7.3	Class-Specific AS Valid Frequency Range Descriptor.....	100
4.8	AudioStreaming Endpoint Descriptors.....	101
4.8.1	AS Isochronous Audio Data Endpoint Descriptors	101
4.8.2	AS Isochronous Feedback Endpoint Descriptor	102
4.9	Class-specific String descriptors	103
5	Requests.....	105
5.1	Standard Requests	105

5.2	Class-Specific Requests	105
5.2.1	AudioControl Requests	106
5.2.2	AudioStreaming Requests	130
5.2.3	Additional Requests	132
6	Interrupts	135
6.1	Interrupt Data Message	135
6.2	Interrupt Sources	137
Appendix A.	Audio Device Class Codes	138
A.1	Audio Function Class Code	138
A.2	Audio Function Subclass Codes	138
A.3	Audio Function Protocol Codes	138
A.4	Audio Interface Class Code	138
A.5	Audio Interface Subclass Codes	139
A.6	Audio Interface Protocol Codes	139
A.7	Audio Function Category Codes	139
A.8	Audio Class-Specific Descriptor Types	140
A.9	Cluster Descriptor Subtypes	140
A.10	Cluster Descriptor Segment types	140
A.11	Channel Purpose Definitions	141
A.12	Channel Relationship Definitions	141
A.13	Ambisonic Component Ordering Convention Types	143
A.14	Ambisonic Normalization Types	143
A.15	Audio Class-Specific AC Interface Descriptor Subtypes	144
A.16	Audio Class-Specific AS Interface Descriptor Subtypes	144
A.17	Audio Class-Specific String descriptor Subtypes	144
A.18	Extended Terminal Segment Types	145
A.19	Effect Unit Effect Types	145
A.20	Processing Unit Process Types	145
A.21	Audio Class-Specific Endpoint Descriptor Subtypes	146
A.22	Audio Class-Specific Request Codes	146
A.23	Control Selector Codes	146
A.23.1	AudioControl Interface Control Selectors	146
A.23.2	Clock Source Control Selectors	146
A.23.3	Clock Selector Control Selectors	146
A.23.4	Clock Multiplier Control Selectors	147
A.23.5	Terminal Control Selectors	147
A.23.6	Mixer Control Selectors	147
A.23.7	Selector Control Selectors	147
A.23.8	Feature Unit Control Selectors	148
A.23.9	Effect Unit Control Selectors	148
A.23.10	Processing Unit Control Selectors	150

A.23.11	Extension Unit Control Selectors.....	150
A.23.12	AudioStreaming Interface Control Selectors.....	150
A.23.13	Endpoint Control Selectors	151
A.24	Connector Types	151

IECNORM.COM : Click to view the full PDF of IEC 62680-1-5:2019

LIST OF TABLES

Table 4-1: Traditional Class-Specific Descriptor Layout	53
Table 4-2: High Capability Class-Specific Descriptor Layout	54
Table 4-3: Cluster Descriptor Header	55
Table 4-4: Cluster Descriptor Segment	56
Table 4-5: End Segment	57
Table 4-6: Cluster Description Segment.....	57
Table 4-7: Vendor-defined Segment	57
Table 4-8: Channel Relationships	58
Table 4-9: Information Segment	62
Table 4-10: Ambisonic Segment.....	62
Table 4-11: Channel Description Segment.....	63
Table 4-12: Vendor-defined Segment	63
Table 4-13: Cluster Descriptor Example.....	63
Table 4-14: Standard AC Interface Descriptor	67
Table 4-15: Class-Specific AC Interface Header Descriptor.....	68
Table 4-16: Input Terminal Descriptor	70
Table 4-17: Output Terminal Descriptor	71
Table 4-18: Extended Terminal Descriptor Header.....	72
Table 4-19: Cluster Descriptor Segment	73
Table 4-20: End Segment	73
Table 4-21: Vendor-defined Segment.....	74
Table 4-22: Bandwidth Segment	74
Table 4-23: Magnitude Segment.....	75
Table 4-24: Magnitude/Phase Segment.....	75
Table 4-25: Position Segment	76
Table 4-26: Position Segment	76
Table 4-27: Vendor-defined Segment.....	76
Table 4-28: Connectors Descriptor	77
Table 4-29: Mixer Unit Descriptor.....	81
Table 4-30: Selector Unit Descriptor	82
Table 4-31: Feature Unit Descriptor	83
Table 4-32: Sampling Rate Converter Unit Descriptor	84
Table 4-33: Effect Unit Descriptor.....	85
Table 4-34: Parametric Equalizer Section Effect Unit Descriptor.....	86
Table 4-35: Reverberation Effect Unit Descriptor.....	87
Table 4-36: Modulation Delay Effect Unit Descriptor	88
Table 4-37: Dynamic Range Compressor Effect Unit Descriptor.....	89
Table 4-38: Common Part of the Processing Unit Descriptor	90
Table 4-39: Up/Down-mix Processing Unit Descriptor	91

Table 4-40: Stereo Extender Processing Unit Descriptor	92
Table 4-41: Multi-Function Processing Unit Descriptor	93
Table 4-42: Extension Unit Descriptor	94
Table 4-43: Clock Source Descriptor	95
Table 4-44: Clock Selector Descriptor	96
Table 4-45: Clock Multiplier Descriptor	96
Table 4-46: Power Domain Descriptor	97
Table 4-47: Standard AC Interrupt Endpoint Descriptor	98
Table 4-48: Standard AS Interface Descriptor	99
Table 4-49: Class-Specific AS Interface Descriptor	100
Table 4-50: Class-Specific AS Valid Frequency Range Descriptor	100
Table 4-51: Standard AS Isochronous Audio Data Endpoint Descriptor	101
Table 4-52: Class-Specific AS Isochronous Audio Data Endpoint Descriptor	102
Table 4-53: Standard AS Isochronous Feedback Endpoint Descriptor	103
Table 4-54: Class-specific String Descriptor	104
Table 5-1: Request Layout	107
Table 5-2: 1-byte Control CUR Parameter Block	108
Table 5-3: 1-byte Control RANGE Parameter Block	109
Table 5-4: 2-byte Control CUR Parameter Block	109
Table 5-5: 2-byte Control RANGE Parameter Block	109
Table 5-6: 4-byte Control CUR Parameter Block	110
Table 5-7: 4-byte Control RANGE Parameter Block	110
Table 5-8: INTEN Parameter Block	110
Table 5-9: Insertion Control CUR Parameter Block	113
Table 5-10: Band Numbers and Center Frequencies (ANSI S1.11-1986 Standard)	117
Table 5-11: Graphic Equalizer Control CUR Parameter Block	118
Table 5-12: Graphic Equalizer Control RANGE Parameter Block	118
Table 5-13: Valid Alternate Settings Control CUR Parameter Block	131
Table 5-14: Audio Data Format Control CUR Parameter Block	131
Table 5-15: Memory Request Values	132
Table 5-16: String Request	133
Table 5-17: High Capability Descriptor Request	134
Table 6-1: Interrupt Data Message Format	136
Table A-1: Audio Function Class Code	138
Table A-2: Audio Function Subclass Codes	138
Table A-3: Audio Function Protocol Codes	138
Table A-4: Audio Interface Class Code	138
Table A-5: Audio Interface Subclass Codes	139
Table A-6: Audio Interface Protocol Codes	139
Table A-7: Audio Function Category Codes	139
Table A-8: Audio Class-specific Descriptor Types	140

Table A-9: Audio Class-Specific Cluster Descriptor Subtypes	140
Table A-10: Cluster Descriptor Segment Types	140
Table A-11: Channel Purpose Definitions	141
Table A-12: Channel Relationship Definitions	141
Table A-13: Ambisonic Component Ordering Convention Types	143
Table A-14: Ambisonic Normalization Types	143
Table A-15: Audio Class-Specific AC Interface Descriptor Subtypes	144
Table A-16: Audio Class-Specific AS Interface Descriptor Subtypes	144
Table A-17: Audio Class-Specific String descriptor Subtypes	144
Table A-18: Extended Terminal Segment Types	145
Table A-19: Effect Unit Effect Types	145
Table A-20: Processing Unit Process Types	145
Table A-21: Audio Class-Specific Endpoint Descriptor Subtypes	146
Table A-22: Audio Class-Specific Request Codes	146
Table A-23: AudioControl Interface Control Selectors	146
Table A-24: Clock Source Control Selectors	146
Table A-25: Clock Selector Control Selectors	146
Table A-26: Clock Multiplier Control Selectors	147
Table A-27: Terminal Control Selectors	147
Table A-28: Mixer Control Selectors	147
Table A-29: Selector Control Selectors	147
Table A-30: Feature Unit Control Selectors	148
Table A-31: Reverberation Effect Unit Control Selectors	148
Table A-32: Reverberation Effect Unit Control Selectors	149
Table A-33: Modulation Delay Effect Unit Control Selectors	149
Table A-34: Dynamic Range Compressor Effect Unit Control Selectors	149
Table A-35: Up/Down-mix Processing Unit Control Selectors	150
Table A-36: Stereo Extender Processing Unit Control Selectors	150
Table A-37: Extension Unit Control Selectors	150
Table A-38: AudioStreaming Interface Control Selectors	150
Table A-39: Endpoint Control Selectors	151
Table A-40: Connector Types	151

LIST OF FIGURES

Figure 3-1: Multiple Configurations and their Audio Interface Associations and Interfaces	21
Figure 3-2: Audio Function Global View	22
Figure 3-3: Inside the Audio Function	32
Figure 3-4: Input Terminal Icon	34
Figure 3-5: Output Terminal Icon	35
Figure 3-6: Mixer Unit Icon	35

Figure 3-7: Selector Unit Icon.....	36
Figure 3-8: Feature Unit Icon	36
Figure 3-9: Sampling Rate Converter Unit Icon.....	37
Figure 3-10: PEQS Effect Unit Icon	38
Figure 3-11: Reverberation Effect Unit Icon	39
Figure 3-12: Modulation Delay Effect Unit Icon.....	39
Figure 3-13: Dynamic Range Compressor Transfer Characteristic.....	40
Figure 3-14: Dynamic Range Compressor Effect Unit Icon	41
Figure 3-15: Up/Down-mix Processing Unit Icon	42
Figure 3-16: Stereo Extender Processing Unit Icon.....	42
Figure 3-17 Multi-Function Processing Unit Icon.....	43
Figure 3-18: Extension Unit Icon	44
Figure 3-19: Clock Source Icon	45
Figure 3-20: Clock Selector Icon.....	45
Figure 3-21: Clock Multiplier Icon	45
Figure 4-1: Cluster Descriptor	55
Figure 4-2: Cluster Descriptor Block.....	56
Figure 4-3: 3D Representation of the Channel Relationships	61
Figure 4-4: Physical to Logical Cluster Mapping.....	66
Figure 4-5: Logical to Physical Cluster Mapping.....	67
Figure 4-6: Extended Terminal Descriptor	72
Figure 4-7: Extended Terminal Channel Block	73
Figure 4-8: Mixer internals.....	80

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.

1.2 PURPOSE

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

1.3 RELATED DOCUMENTS

- *Universal Serial Bus Specification*, Revision 2.0 (referred to in this document as the *USB Specification*). In particular, see Chapter 5, “USB Data Flow Model” and Chapter 9, “USB Device Framework.”
- *Universal Serial Bus 3.1 Specification*, Revision 1.0 (referred to in this document as the *USB 3.1 Specification*). This document covers details specific to SuperSpeed and SuperSpeed+ devices.
- *Universal Serial Bus Device Class Definition for Audio Data Formats* (referred to in this document as USB Audio Data Formats).
- *Universal Serial Bus Device Class Definition for Terminal Types* (referred to in this document as USB Audio Terminal Types).
- ANSI S1.11-1986 standard.
- MPEG-1 standard ISO/IEC 111172-3 1993.
- MPEG-2 standard ISO/IEC 13818-3 Feb. 20, 1997.
- Digital Audio Compression Standard (AC-3), ATSC A/52A Aug. 20, 2001. (available from <http://www.atsc.org/>)
- ANSI/IEEE-754 floating-point standard.
- ISO/IEC 60958 International Standard: *Digital Audio Interface and Annexes*.
- ISO/IEC 61937 standard.

1.4 TERMS AND ABBREVIATIONS

This section defines terms used throughout this document. For additional terms that pertain to the Universal Serial Bus, see Chapter 2, “Terms and Abbreviations,” in the *USB Specification*.

ASRC	Acronym for Asynchronous Sampling Rate Conversion.
(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 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.

AEC	Acronym for Audio Echo Cancellation.
Audio Function	Independent part of a USB device that deals with audio-related functionality.
Audio Interface Association (AIA)	Grouping of a single AudioControl interface, zero or more AudioStreaming interfaces and zero or more MIDISTreaming interfaces that together constitute a complete interface to a particular version (compliance level) of the 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.
Clock Multiplier (CM)	Multiplies a clock signal by a fractional multiplier.
Clock Selector (CX)	Selects from a number of input clock signals.
Clock Source (CS)	Generates an audio-related clock signal (master clock or sample clock).
CMD	Acronym for Clock Multiplier descriptor.
CSD	Acronym for Clock Source descriptor.
CXD	Acronym for Clock Selector descriptor.
Effect Unit (EU)	Provides advanced audio manipulation on the incoming logical audio channels.
Entity	Addressable logical object inside an Audio Function.
Extension Unit (XU)	Applies an undefined process to a number of logical input channels.
Feature Unit (FU)	Provides basic audio manipulation on the incoming logical audio channels.
FUD	Acronym for Feature Unit descriptor.
High Capability Descriptor	A new representation for class-specific descriptors that allows for potentially large (>256 bytes) and dynamic descriptors.
Input Pin	Logical input connection to an Entity. Carries a single Cluster.
Input Terminal (IT)	Receptacle for audio information flowing into the Audio Function.
ITD	Acronym for Input Terminal descriptor.
Logical Audio Channel	Logical transport medium for a single audio channel. Makes abstraction of the physical properties and formats of the connection. Is usually identified by spatial location. Examples are Left channel, Right Surround channel, etc.
MIDISTreaming interface (MSI)	USB interface that may be used to transport MIDI data streams into or out of the Audio Function.
Mixer Unit (MU)	Mixes a number of logical input channels into a number of logical output channels.

MUD	Acronym for Mixer Unit descriptor.
OTD	Acronym for Output Terminal descriptor.
Output Pin	Logical output connection to an Entity. Carries a single Cluster.
Output Terminal (OT)	An outlet for audio information flowing out of the Audio Function.
Phase Locked Loop	A hardware or software control system that generates an output signal whose phase is related to the phase of an input signal. Typically used in clock recovery applications.
Processing Unit (PU)	Applies a predefined process to a number of logical input channels.
PUD	Acronym for Processing Unit descriptor.
Reserved	Reserved is a keyword indicating reserved bits, bytes, words, fields, and code values that are set-aside for future standardization. Their use and interpretation may be specified by future extensions to this specification and, unless otherwise stated, shall not be utilized or adapted by vendor implementation. A reserved bit, byte, word, or field shall be set to zero by the sender and shall be ignored by the receiver.
RUD	Acronym for Sampling Rate Converter Unit descriptor.
Sampling Rate Converter Unit (RU)	Converts the incoming audio data stream, running at a sampling rate, synchronous to a first master clock, into an outgoing audio data stream that is running at a sampling rate, synchronous to a second master clock, which is free running with respect to the first master clock.
Selector Unit (SU)	Selects from a number of input 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 sub-functionality.
XUD	Acronym for Extension Unit descriptor.

2 MANAGEMENT OVERVIEW

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

In principle, a versatile bus specification like the USB provides many ways to propagate and/or control 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* and the *USB 3.1 Specification*. The Audio Device Class definition adheres to these synchronization schemes to transport audio data reliably over the bus.

This document contains all necessary information for a designer to build a USB-compliant device that incorporates Audio Functionality. It specifies the standard and class-specific descriptors that shall be present in each USB Audio Function. It further explains the use of class-specific requests that allow for full Audio Function control. A number of predefined data formats are listed and fully documented. Each format defines a standard way of transporting audio over the USB. Provisions have been made so that vendor-specific audio formats and compression schemes can be handled as well.

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

2.1 OVERVIEW OF KEY DIFFERENCES BETWEEN ADC V2.0 AND V3.0

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

- Dolby Processing Unit removed
- Encoder and Decoder support removed
- Copy protection (S/PDIF-style) removed
- bmAttributes (MaxPacketsOnly bit) field in class-specific isochronous endpoint descriptor removed
- Type II and Extended Type II Audio Data Formats removed
- New Power Domains
- New Multi-Function Processing Unit
- Additional Type III formats introduced
- New Cluster descriptor
- New Extended Terminal descriptor

- New Connectors descriptor
- Layout of all class-specific descriptors has changed
- New class-specific String descriptors
- New High Capability descriptors
- Additional sources for interrupts
- Provides for backward compatibility, using multiple Configuration descriptors
- Support for LPM/L1
- Mandated support for BADD on each ADC3.0 compliant device

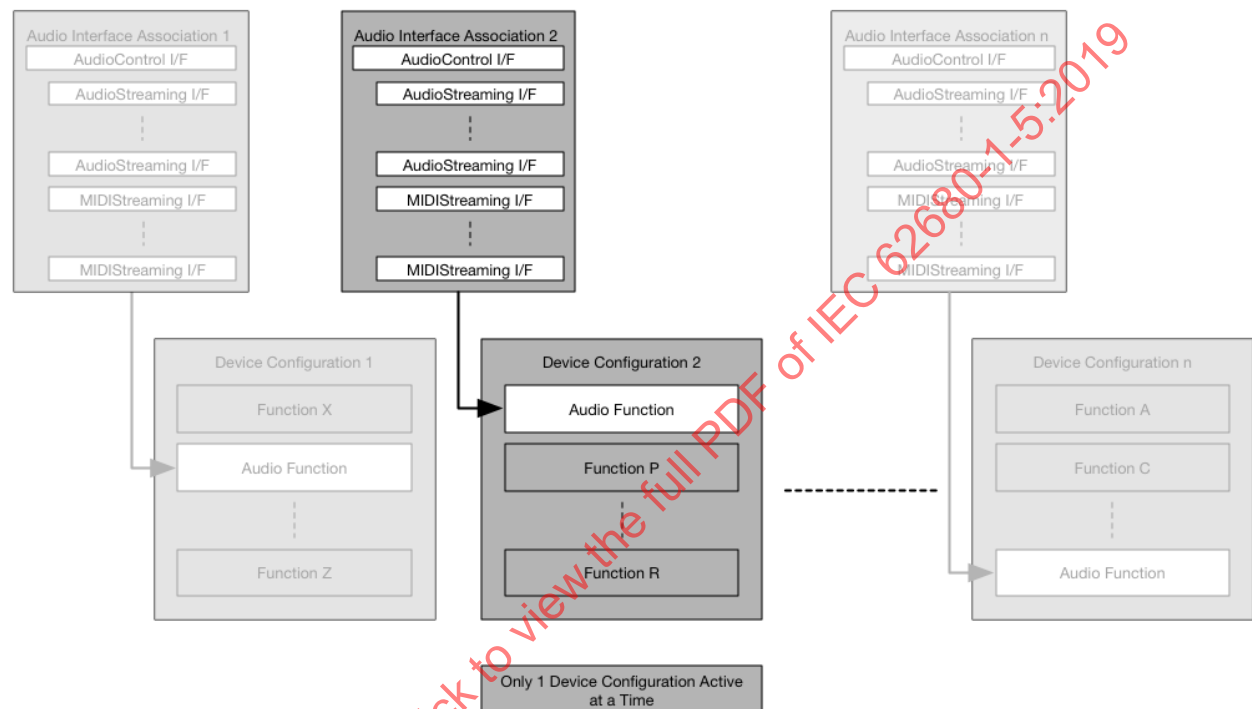
IECNORM.COM : Click to view the full PDF of IEC 62680-1-5:2019

3 FUNCTIONAL CHARACTERISTICS

3.1 INTRODUCTION

In many cases, Audio Functionality does not exist as a standalone device. It is one capability that, together with other functions, constitutes a “composite” device. A perfect example of this is a DVD-ROM player, which can incorporate video, audio, data storage, and transport control. The Audio Function is thus located at the interface level in the device class hierarchy. The following figure provides details.

Figure 3-1: Multiple Configurations and their Audio Interface Associations and Interfaces



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

A USB Device may express different AIAs representing the same underlying hardware by using multiple USB Device Configuration descriptors. These AIAs may be used to provide access to the Audio Function at different compliance levels. For example, one AIA might provide access to the Audio Function in ADC 1.0 mode, whereas another AIA might provide access to the Audio Function in ADC 3.0 mode, as defined by this document. Note that the Audio Function may take on a different topology and even different functionality when accessed through different AIAs.

Because only one Configuration may be active at a time, only one AIA shall be active at a time for the same Audio Function. Host software should choose the most desirable Configuration and use the Standard SET_CONFIGURATION command to select the Configuration containing the most desired AIA or AIAs (if there are multiple independent Audio Functions residing in the same USB Device).

An AIA shall have a single AudioControl interface and can have zero or more AudioStreaming and zero or more MIDIStreaming interfaces. The AudioControl (AC) interface is used to access the Audio Controls of the function whereas the AudioStreaming (AS) interfaces are used to transport audio streams into and out of the Audio Function. MIDIStreaming (MS) interfaces can be used to transport MIDI data streams into and out of the Audio Function.

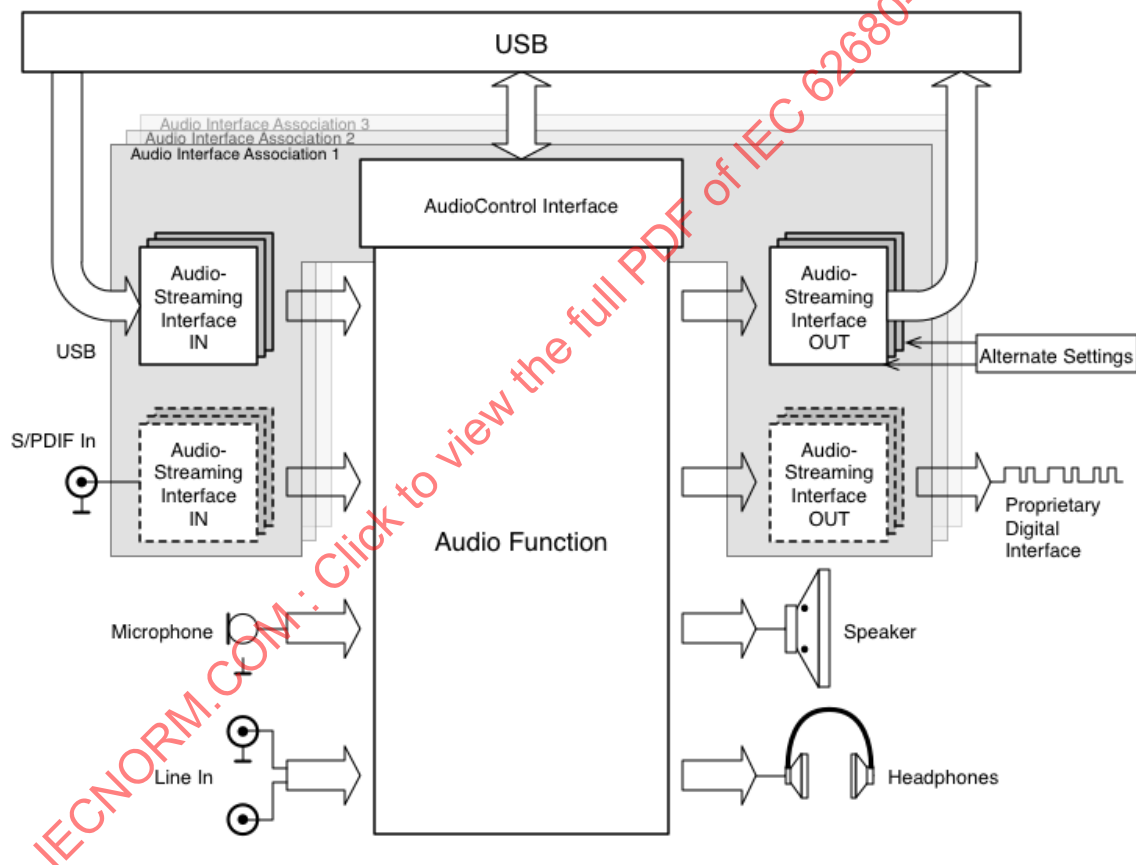
Note: All MIDI-related information is grouped in a separate document, *Universal Serial Bus Device Class Definition for MIDI Devices* that is considered part of this specification. The remainder of this document will therefore not mention MIDIStreaming interfaces and their specifics anymore.

Note that AudioStreaming interfaces do not necessarily all represent USB audio streams. Any audio stream, regardless of its origin or destination that enters or leaves the Audio Function may have an associated AudioStreaming interface, if that interface requires the Host to have access and/or visibility to some interface Controls or descriptors that may have an impact on the operation of the interface from a USB perspective.

A device can have *multiple independent* Audio Functions located in the same composite Device. They are each accessed through their active Audio Interface Association, which must be contained within the currently selected USB Configuration.

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

Figure 3-2: Audio Function Global View



All functionality pertaining to controlling parameters that directly influence audio perception (like volume) are located inside the central rectangle and are exclusively controlled through the AudioControl interface. Streaming aspects of the communication to or from the Audio Function are handled through separate AudioStreaming interfaces, as necessary. Each USB audio stream shall be represented by an AudioStreaming interface. A non-USB audio stream may be represented by an AudioStreaming interface if there is a need to convey information between that audio stream's interface and the Host. All control data that is related specifically to the streaming behavior of the interface is conveyed through the AudioStreaming interface.

A physical S/PDIF connection to the Audio Function is a typical example of a non-USB AudioStreaming interface. Although the actual audio data is coming in from the outside world (not through the USB), it might be necessary to control some aspects of the S/PDIF connection. In that case, the S/PDIF connection is represented by an AudioStreaming interface so that it becomes addressable through USB.

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

3.2 BASIC AUDIO DEVICE DEFINITION

The Basic Audio Device Definition (BADD) exists as a separate document and is based on and fully compatible with this Audio Device Class Definition (ADC 3.0). As such, the BADD defines a subset of functionality commonly found in Headphones, Microphones, and Headsets using the features and tools provided by the ADC but it removes all of the optionality allowed by the ADC and prescribes rigorously how certain features shall be implemented. This allows simple Host drivers to know exactly what to expect when encountering a BADD-compliant device, reducing significantly the work a driver needs to perform parsing the BADD Device’s descriptors and interacting with the Device.

3.3 BACKWARDS COMPATIBILITY

This 3.0 version of the *Audio Device Class Definition* (ADC 3.0) as such is not backwards compatible with any of the previous versions of the same specification.

However, to ensure the broadest possible interoperability among new ADC 3.0 Devices and legacy ADC 1.0 or ADC 2.0 Hosts, this specification requires that the ADC 3.0 compliant Device shall do the following:

- The first USB Configuration descriptor (index 0) exposed by the Device shall contain an Audio Interface Association (AIA), which is compliant with ADC 1.0 or ADC 2.0. In other words, if Host software selects the first indexed Device Configuration, it shall expose an Audio Function that is compliant with ADC 1.0 or ADC 2.0. This requirement should allow the Audio Function to interoperate with existing Hosts exactly the same as current legacy ADC 1.0 or ADC 2.0 Functions.
- If the Audio Function contains at least one USB Audio Streaming Interface, the Device shall have another Configuration descriptor that exposes an AIA which is BADD 3.0 compliant. A Device shall only have a single AIA of this type. However, this same BADD compliant AIA may need to be duplicated in more than one Configuration descriptor. This requirement guarantees that the Device will interoperate with new ADC 3.0 Hosts that support the BADD 3.0 specification. To activate this mode of operation for the Audio Function, the Host needs to select a Device Configuration which contains the BADD 3.0 compliant AIA.
- The Device may have one or more Configurations that expose one or more AIAs which shall be compliant with ADC 3.0. This way the Audio Function may provide functionality beyond what is available in BADD 3.0 operating mode. To activate this mode of operation, the Host needs to explicitly select a Configuration that contains the desired AIA(s).

It is strongly recommended that ALL ADC 3.0 Hosts support the BADD 3.0 specification.

3.4 AUDIO INTERFACE ASSOCIATION (AIA) AND INTERFACE ASSOCIATION DESCRIPTOR

On the USB, an Audio Function is completely defined by its interfaces. An Audio Function has one or more Audio Interface Associations (AIA) that provide access to it at different compliance levels. Only one AIA can be active at one time. Each AIA shall have one AudioControl interface and can have zero or more AudioStreaming interfaces.

The standard USB Interface Association mechanism is used to describe the Audio Interface Association i.e. to bind those interfaces together. Interface Association is expressed via the standard USB Interface Association descriptor (IAD). Every Interface Association descriptor has a **bFunctionClass**, **bFunctionSubClass** and **bFunctionProtocol** field that together identify the function that is represented by the Association. The following paragraphs define these fields for the Audio Device Class.

3.4.1 AUDIO FUNCTION CLASS

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

The Audio Function Class code is assigned by the USB-IF. For details, see Appendix A.1, “Audio Function Class Code”.

3.4.2 AUDIO FUNCTION SUBCLASS

The Audio Function Subclass is contained in the **bFunctionSubClass** field of a Standard Interface Association descriptor. Any AIA exposed by a Device that is compliant with this specification shall use this field to further define the device. This field indicates whether the AIA is a full-fledged implementation of the ADC 3.0 specification or is compliant with one of the BADD-defined Profiles.

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

3.4.3 AUDIO FUNCTION PROTOCOL

The Audio Function Protocol is contained in the **bFunctionProtocol** field of a Standard Interface Association descriptor. The Function Protocol code is used to reflect the compliance level of the Audio Function so that enumeration software can decide which driver version needs to be instantiated. This specification requires that the Function Protocol code be the same as the Audio Interface (both AudioControl and AudioStreaming) Protocol code.

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

3.5 AUDIO INTERFACE CLASS

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

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

3.6 AUDIO INTERFACE SUBCLASS

The Audio Interface Subclass code is contained in the **bInterfaceSubClass** field of the Standard Interface descriptor of any Interface contained in an AIA. All USB interfaces that provide functionality through this specification, as well as any interfaces that provide MIDI streaming functionality, shall use one of the following Audio Interface Subclass codes as provided by this specification:

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

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

3.7 AUDIO INTERFACE PROTOCOL

The Audio Interface Protocol code is contained in the **InterfaceProtocol** field of the Standard Interface descriptor of any Interface contained in an AIA. The Interface Protocol code is used to reflect the compliance level of the Audio Function.

All Audio Functions compliant with this ADC 3.0 specification shall use Interface Protocol Code IP_VERSION_03_00. The assigned codes can be found in Appendix A.6, “Audio Interface Protocol Codes” of this specification. All other Protocol codes are unused and reserved by this specification for future use.

3.8 AUDIO FUNCTION CATEGORY

The Audio Function Category code is contained in the Class Specific Audio Control Interface descriptor as defined by this specification. This code indicates the primary intended use for the Audio Function. The following Function Categories are currently defined in this specification:

- Desktop Speaker: One or more speakers set up in a small environment to provide audio intended primarily for one person.
- Home Theater: Several speakers set up in a moderately sized environment to provide audio levels significantly louder than a Desktop Speaker setup and intended to be clearly heard by multiple people.
- Microphone: A device set up to record audio from audible sources.
- Headset: A device with at least one speaker and at least one microphone designed to be worn or held by a user to provide personal audio playback and voice input capabilities.
- Telephone: A Headset or handset type device that also connects to a telephone system, (e.g. POTs, PBX, VoIP) capable of making and receiving telephone calls.
- Converter: A device that allows conversion of audio from one electrical or optical format to another electrical or optical format, and/or converting audio data from one encoding format to another (e.g. AC-3 to PCM, etc.).
- Voice/Sound recorder: A device set up with at least one microphone and at least one speaker that is designed to operate, at least some of the time, independently of the Host to record and store audible sources and play back its recorded content.
- IO Box: A device designed to deliver one or more, possibly different, electrical and optical inputs and outputs for connection to other devices.
- Musical Instrument: A musical instrument, e.g. piano, guitar, synthesizer, drum machine, etc.
- Pro-Audio: A device not typically used by consumers of audio, e.g. editing equipment, multitrack recording equipment, etc.
- Audio/Video: The audio from a device that also supplies simultaneous video where the expectation is that the audio is tightly coupled to the video, e.g. a camcorder, a DVD player, a television, etc.
- Control Panel: A device that is used to control the flow of audio through a system of audio devices, such as a mixer panel.
- Other: Any device whose primary purpose is sufficiently different from the above descriptions as to be considered a completely different form of device.

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

3.9 CLOCK DOMAINS

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

Multiple different Clock Domains may exist within the same Audio Function.

3.10 POWER DOMAINS

A Power Domain is defined as a zone within the Audio Function that groups one or more Entities and allows the Host to control power consumption levels for that Domain. This way, the Host can potentially switch parts of the Audio Function to lower power states when these parts are currently not in use, leading to an overall decrease in power consumption. As an example, a USB headset may have two separate Power Domains, one for the output-related functionality (headphone) and one for the input-related functionality (microphones). When the headset is used to simply listen to music, the input-related functionality may be temporarily switched to a low power state (or even completely off) to conserve power. This is especially important if the headset is used in conjunction with a battery-powered device, such as a mobile phone or MP3 player, where extending battery life is important.

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

3.11 AUDIO SYNCHRONIZATION TYPES

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

3.11.1 ASYNCHRONOUS

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

3.11.2 SYNCHRONOUS

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

3.11.3 ADAPTIVE

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

3.11.4 IMPLICATIONS OF THE DIFFERENT SYNCHRONIZATION TYPES

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

3.11.4.1 SINGLE DIRECTION SINK ENDPOINT

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

- Asynchronous
 - **Host:** The Host driver needs to be able to handle an explicit feedback endpoint. From the feedback data, the Host then decides how many samples to send over the data streaming endpoint in subsequent Bus Intervals.
 - **Device:** The Device has its own local, free-running audio sample clock, which determines how many samples are consumed by the Device each Bus Interval. The Device shall implement an explicit feedback endpoint as well as the necessary logic to provide the correct feedback values to send over said endpoint back to the Host. The advantage of this mode of operation is that it is fairly easy to generate a robust, stable, jitter-free, high-quality audio sample clock (derived from a crystal-based master clock, for example).
- Synchronous
 - **Host:** The Host needs to send out a known number of bytes for each packet going to the Device. The Host may need to generate a (fixed) pattern of audio samples to achieve the desired sampling rate. As an example, to generate a sampling rate of 44.1 kHz in a Full-Speed implementation, the Host needs to send a repeating pattern of nine packets containing 44 audio samples, followed by one packet containing 45 audio samples.
 - **Device:** This synchronization type requires the Device to implement either an audio clock PLL or an ASRC.
- Adaptive
 - **Host:** The Host may use any method or means to determine how many samples per Bus Interval to transmit. Effectively operating as a “Synchronous-to-SOF” Source is an easy approach, but not the only one allowed by the USB specification.
 - **Device:** This synchronization type requires the Device to implement either an audio clock PLL or an ASRC to adapt to the average number of samples arriving over a certain period of time.

3.11.4.2 SINGLE DIRECTION SOURCE ENDPOINT

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

- Asynchronous
 - **Host:** The Host needs to operate as an Adaptive Sink. Depending on the Host’s system design, this may require an ASRC or an audio clock PLL on the Host side.
 - **Device:** The Device has its own local, free-running audio sample clock, which determines how many samples are produced by the Device each Bus Interval. The advantage of this mode of operation is that it is fairly easy to generate a robust, stable, jitter-free, high-quality audio sample clock (derived from a crystal-based master clock, for example).
- Synchronous
 - **Host:** The Host receives a known number of audio samples in each packet coming from the Device. The Host should expect a fixed pattern of audio samples to achieve the desired sampling rate.

- **Device:** This synchronization type requires the Device to implement either an audio clock PLL to lock on to the SOF or the start of Bus Interval and generate a high-quality audio sample clock directly or use an ASRC as a bridge between the local and USB clock domains. The Device shall generate a fixed packet size pattern as described in the *USB Audio Data Formats Definition* document.
- Adaptive
 - **Host:** This scenario requires the implementation of a feedforward OUT endpoint in the Device that allows the Host to inform the Device how many samples per Bus Interval to send to the Host over the streaming data IN endpoint.
 - **Device:** This synchronization type requires the Device to implement an audio clock PLL or an ASRC to adapt to the sample rate being communicated by the Host via the feedforward OUT endpoint.

3.11.4.3 SOURCE AND SINK ENDPOINTS

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

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

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

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

Note: Using either of these two implicit feedback mechanisms would preclude the ability of an ADC 3.0 compliant device from being able to use Power Domains to shut down either the source or the sink, as data must remain flowing in both directions to provide the feedback/feedforward information.

3.12 INTER CHANNEL SYNCHRONIZATION

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

3.13 AUDIO FUNCTION TOPOLOGY

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

Units provide the basic building blocks to fully describe most Audio Functions. Audio Functions are built by connecting together several of these Units. A Unit has one or more Input Pins and a single Output Pin, where each Pin represents a Cluster of logical audio channels inside the Audio Function (see Section 3.13.1, “Cluster”).

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

In addition, the concept of a Terminal is introduced. There are two types of Terminals. An Input Terminal (IT) is an Entity that represents a starting point for audio channels inside the Audio Function. An Output Terminal (OT) represents an ending point for audio channels inside the Audio Function. From the Audio Function's perspective, a USB endpoint is a typical example of an Input or Output Terminal. It either provides data streams to the Audio Function (IT) or consumes data streams coming from the Audio Function (OT). Likewise, a Digital to Analog converter, built into the Audio Function is represented as an Output Terminal in the Audio Function's model. Connection to the Terminal is made through its single Input or Output Pin.

Input Pins of a Unit are numbered starting from one up to the total number of Input Pins on the Unit. The Output Pin number is always one. Input Terminals have only one Output Pin and its number is always one. Output Terminals have only one Input Pin and it is always numbered one.

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

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

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

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

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

- Clock Source (CS)
- Clock Selector (CX)
- Clock Multiplier (CM)

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

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

A Clock Selector is used to select between multiple sampling clock signals that might be available inside an Audio Function. It has multiple Clock Input Pins and a single Clock Output pin. Clock Input Pins are numbered

starting from one up to the total number of Clock Input Pins on the Clock Selector. The Clock Output Pin number is always one.

A Clock Multiplier is used to derive a new clock signal with a different frequency from the clock signal at its single Clock Input Pin. It does this by multiplying that clock signal frequency by a numerator P and then dividing it by a denominator Q. The new clock signal is guaranteed to be synchronous with the input clock signal. A Clock Multiplier has one Input Pin and one Output Pin and their numbers are always one. An implementation can choose to make the values P and Q programmable, although most Audio Functions will expose P and Q Controls as Read-Only and report any changes to those values via the interrupt mechanism. Such changes may be necessary when the Audio Function detects a change in native sampling rate for incoming encoded content, for example.

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

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

Each Input and Output Terminal has a single Clock Input Pin that is connected to a Clock Output Pin of a Clock Entity. The clock signal carried by that Clock Output Pin determines at which sampling frequency the hardware represented by the Terminal is operating.

Each Sampling Rate Converter Unit has two Clock Input Pins that are typically connected to the Clock Output Pins of two different Clock Entities. The clock signals carried by those Clock Output Pins determine the sampling frequencies between which the Sampling Rate Converter Unit is converting.

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

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

The ensemble of Unit descriptors, Terminal descriptors and Clock Entity descriptors provide a full description of the Audio Function to the Host. This information is typically retrieved from the device at enumeration time. By parsing the descriptors, a generic audio driver should be able to fully control the Audio Function, except for the functionality represented by Extension Units. Those require vendor-specific extensions to the audio class driver.

Important Note:

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

Figure 3-3, “Inside the Audio Function” illustrates the concepts defined above. Using the iconic symbols defined further, it describes a hypothetical Audio Function that incorporates 15 Entities: three Input Terminals, five Units, three Output Terminals, two Clock Sources, a Clock Selector, and one Clock Multiplier. Each Entity has its unique ID (from 1 to 15) and descriptor that fully describes the functionality of the Entity and also how that particular Entity is connected into the overall topology of the Audio Function.

Input Terminal 1 (IT 1) is the representation of a USB OUT endpoint used to stream audio from the Host to the audio device. IT 2 is the representation of an analog Line-In connector on the audio device whereas IT 3 is an analog Microphone-In connector on the audio device. Selector Unit 4 (SU 4) selects between the audio coming from the Host and the audio present at the Line-In connector. Feature Unit 5 (FU 5) is then used to manipulate

the audio (Volume, Bass, Treble ...) before it is presented to Output Terminal 9 (OT 9). OT 9 is the representation of a Headphone Out jack on the audio device.

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

Clock Source 12 (CS 12) represents an internal sampling frequency generator, running at 96 kHz for instance. Clock Source 15 (CS 15) is the representation of an external master sampling clock input that can be used to synchronize the device to an external source. Clock Selector 13 (CS 13) enables selection between the two available Clock Sources. The output of CS 13 provides a sampling frequency to IT 1, IT 2, IT3, OT 10, and OT 11 of 96 kHz. Clock Multiplier CM 14 further multiplies that clock signal by 0.5, providing a sampling frequency of 48 kHz to OT 9 for driving the headphone. Since all sampling frequencies used inside the Audio Function are at all times derived from a single master clock (internal or external), all audio streams in the Audio Function are synchronous.

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

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

The diagram illustrates the system architecture with the following components and connections:

- Inputs:**
 - USB OUT:** Connected to IT (1), which links to IT Descr.
 - Analog Line IN:** Connected to IT (2), which links to IT Descr.
 - Analog Mic IN:** Connected to IT (3), which links to IT Descr.
- Processing Path:**
 - The IT units connect to a **Mixer Unit** (6) and a **Selector Unit** (4).
 - The **Mixer Unit** (6) outputs to a **Processing Unit** (7), which links to PU Descr.
 - The **Processing Unit** (7) outputs to a **Feature Unit** (8), which links to FU Descr.
 - The **Selector Unit** (4) outputs to a **Feature Unit** (5), which links to FU Descr.
- Outputs:**
 - The **Feature Unit** (8) connects to OT (11), which links to OT Descr and **USB IN**.
 - The **Feature Unit** (5) connects to OT (9), which links to OT Descr and **Headphone OUT**.
 - The **Feature Unit** (8) also connects to OT (10), which links to OT Descr and **Speakers**.
- Control Path:**
 - The IT units (1, 2, 3) also connect to a **Control Signal** (CS) block (12), which links to CS Descr.
 - The CS block (12) connects to a **Control Signal** (CX) block (13), which links to CX Descr.
 - The CX block (13) connects to a **Control Signal** (P/Q) block (14), which links to CM Descr.
 - The P/Q block (14) connects to the OT units (9, 10, 11).

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

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

3.13.1 CLUSTER

A Cluster is a grouping of audio channels that carry tightly related synchronous audio information. Inside the Audio Function, complete abstraction is made of the actual physical representation form of the audio data that travels over the connections among Terminals and Units. Each audio channel in the Cluster is considered to be a logical channel and all the physical attributes of the channel (bit width, bit resolution, etc.) are not specified and considered irrelevant in the context of the Audio Function as seen through the AudioControl interface. The fact that an Input Pin and an Output Pin are connected together in the Audio Function's topology is a guarantee (by construction) that the information flowing over the connection is compatible with both Entities that are connected.

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

A Cluster is characterized by a number of attributes:

- A unique identifier for the Cluster
- The number of audio channels in the Cluster
- The purpose of each audio channel, such as Voice, Speech, etc.
- The audio channel relationship (or spatial location) of each audio channel in the Cluster
- An indication whether an audio channel is part of an Ambisonic group

There is a Cluster descriptor (CD) associated with each Cluster that fully describes the Cluster.

There are two types of Clusters used in this specification:

- A logical Cluster describes audio channels within the Audio Function (the closed box) where audio channels are treated as logical concepts.
- A physical Cluster describes physical audio channels, for example when travelling over a connector.

3.13.2 INPUT TERMINAL

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

An Input Terminal can represent inputs to the Audio Function other than USB OUT endpoints. A Line-In connector on an audio device is an example of such a non-USB input. However, if the audio stream is entering the Audio Function by means of a USB OUT endpoint, there is a one-to-one relationship between the AudioStreaming interface that contains this endpoint and its associated Input Terminal. The class-specific interface descriptor contains a field that holds a direct reference to this Input Terminal. The Host needs to use both the AudioStreaming interface and endpoint descriptors and the Input Terminal descriptor to get a full understanding of the characteristics and capabilities of the Input Terminal. Stream-related parameters are stored in the AudioStreaming interface descriptors. Control-related parameters are stored in the Terminal descriptor.

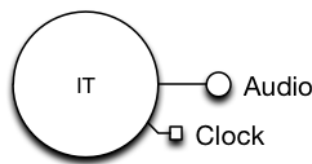
The conversion process from incoming, possibly encoded, audio streams to logical audio channels always involves some kind of decoding engine. The decoding types range from rather trivial decoding schemes like converting interleaved stereo 16 bit PCM data into a Left and Right logical channel to very sophisticated

schemes like converting an MPEG-2 7.1 encoded audio stream into Front Left, Front Left of Center, Front Center, Front Right of Center, Front Right, Back Left, Back Right and Low Frequency Effects logical channels. The decoding engine is considered part of the Entity that actually receives the encoded audio data streams (like a USB AudioStreaming interface). The type of decoding is therefore implied by the value in the **bmFormats** field, located in the class-specific AudioStreaming interface descriptor. The associated Input Terminal deals with the logical channels after they have been decoded.

An Input Terminal has a single Clock Input Pin. The clock signal present at that Pin is used as the sampling clock for all underlying hardware that is represented by this Input Terminal. There is a field in the Input Terminal descriptor that uniquely identifies the Clock Entity to which the Input Terminal is connected.

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

Figure 3-4: Input Terminal Icon



3.13.3 OUTPUT TERMINAL

The Output Terminal (OT) is used to interface between Units inside the Audio Function and the 'outside world'. It serves as an outlet for audio information, flowing out of the Audio Function. Its function is to represent a sink of outgoing audio data before this data is properly packed from the original separate logical channels into the outgoing audio stream (the encoding process). The Cluster enters the Output Terminal through a single Input Pin.

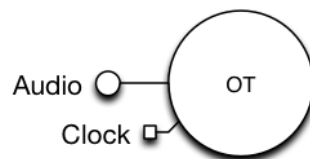
An Output Terminal can represent outputs from the Audio Function other than USB IN endpoints. A speaker built into an audio device or a Line Out connector is an example of such a non-USB output. However, if the audio stream is leaving the Audio Function by means of a USB IN endpoint, there is a one-to-one relationship between the AudioStreaming interface that contains this endpoint and its associated Output Terminal. The class-specific interface descriptor contains a field that holds a direct reference to this Output Terminal. The Host needs to use both the AudioStreaming interface and endpoint descriptors and the Output Terminal descriptor to fully understand the characteristics and capabilities of the Output Terminal. Stream-related parameters are stored in the AudioStreaming interface 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. 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 by the value in the **bmFormats** field, located in the class-specific AudioStreaming interface descriptor. The associated Output Terminal deals with the logical channels before encoding.

An Output Terminal has a single Clock Input Pin. The clock signal present at that Pin is used as the sampling clock for all underlying hardware that is represented by this Output Terminal. There is a field in the Output Terminal descriptor that uniquely identifies the Clock Entity to which the Output Terminal is connected.

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

Figure 3-5: Output Terminal Icon



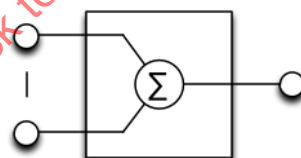
3.13.4 MIXER UNIT

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

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

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

Figure 3-6: Mixer Unit Icon

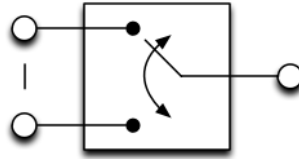


3.13.5 SELECTOR UNIT

The Selector Unit (SU) selects from n Clusters, each containing m logical input channels and routes them unaltered to the single output 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 3-7: Selector Unit Icon



3.13.6 FEATURE UNIT

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

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

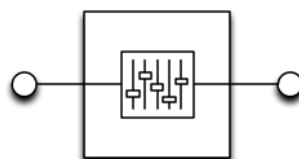
In addition, the Feature Unit optionally provides the above Audio Controls but now influencing all channels of the Cluster at once. In this way, 'master' Controls can be implemented. The master Controls are cascaded after the individual channel Controls. This setup is especially useful in multi-channel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

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

The Feature Unit descriptor reports what Controls are present for every channel in the Feature Unit and for the 'master' channel. All logical channels in a Feature Unit are fully independent. There exist no cross couplings among channels within the Feature Unit. There are as many logical output channels, as there are input channels. These are grouped into one 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 3-8: Feature Unit Icon



3.13.7 SAMPLING RATE CONVERTER UNIT

The Sampling Rate Converter (SRC) Unit (RU) is included here as an optional way to indicate where exactly within the Audio Function sampling rate conversion takes place. In many cases, it is unnecessary to indicate this

point and any SRC Unit can be omitted from the topology without materially affecting the information presented to the Host. The primary reason to include the SRC Unit is to accurately report any latencies incurred by the Sampling Rate Conversion process.

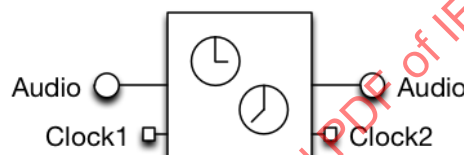
The Sampling Rate Converter Unit provides a bridge function between different clock domains within the Audio Function. The Sampling Rate Converter does not provide any Audio Controls. It takes the audio on all the logical channels in the single input Cluster belonging to a certain clock domain and converts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

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

A SRC Unit has two Clock Input Pins. One Clock Input Pin is associated with the single Input Pin of the SRC Unit. The other Clock Input Pin is associated with the single Output Pin of the SRC Unit. The clock signals present at those two Clock Input Pins identify the two clock domains between which the SRC Unit is converting. Note that it is allowed to have both Clock Input Pins connected to clock signals belonging to the same clock domain.

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

Figure 3-9: Sampling Rate Converter Unit Icon



3.13.8 EFFECT UNIT

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

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

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

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

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

If the Effect Unit is an explicit member of a Power Domain, then switching the Power Domain to any Power State other than D0 shall render the Unit non-functional and its output is undefined. Explicit bypass functionality may be required to preserve the integrity of the signal path.

3.13.8.1 PARAMETRIC EQUALIZER SECTION EFFECT UNIT

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

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

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

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

Figure 3-10: PEQS Effect Unit Icon



3.13.8.2 REVERBERATION EFFECT UNIT

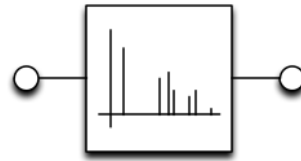
The Reverberation Effect Unit is used to add room acoustics effects to the original audio information. These effects can range from small room reverberation effects to simulation of a large concert hall reverberation. A number of parameters can be manipulated to obtain the desired reverberation effects.

- Reverb Type: Room1, Room2, Room3, Hall1, Hall2, Plate, Delay, and Panning Delay.
- Reverb Level: sets the amount of reverberant sound versus the original sound. Expressed as a ratio.
- Reverb Time: sets the time over which the reverberation will continue. Expressed in s.
- Reverb Delay Feedback: used with Reverb Types Delay and Delay Panning. Sets the way in which delay repeats. Expressed as a ratio.
- Reverb Pre-Delay: sets the delay time between original sound and initial reverb reflection. Expressed in ms.
- Reverb Density: sets the density of the reverb reflections.
- Reverb Hi-Freq Roll-Off: sets the cut-off frequency of a low pass filter on the reflections. Expressed in Hz.

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

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

Figure 3-11: Reverberation Effect Unit Icon



3.13.8.3 MODULATION DELAY EFFECT UNIT

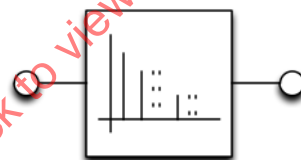
The Modulation Delay Effect Unit is used to add modulation (like chorus) effects to the original audio information. A number of parameters can be manipulated to obtain the desired modulation effects.

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

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

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

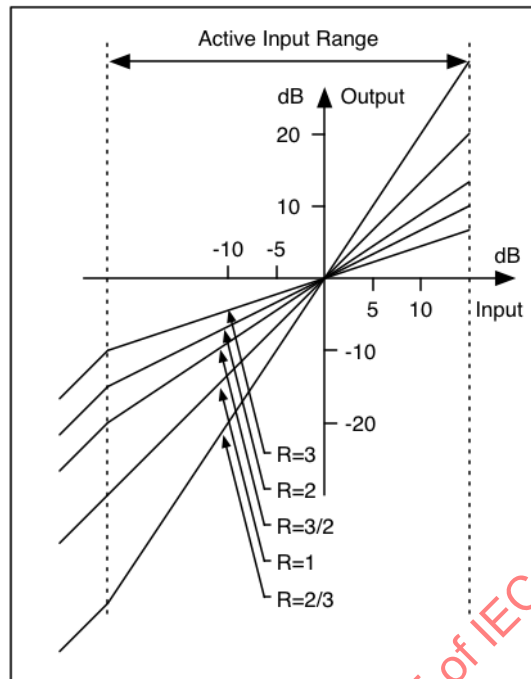
Figure 3-12: Modulation Delay Effect Unit Icon



3.13.8.4 DYNAMIC RANGE COMPRESSOR EFFECT UNIT

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

Figure 3-13: 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.

$$R^{-1} = \frac{\partial \text{Log}(\frac{P_o}{P_R})}{\partial \text{Log}(P_i/P_R)}$$

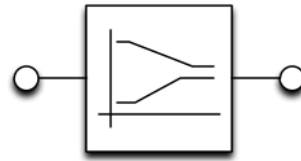
P_R is the reference level and it is made equal to the so-called line level. All levels are expressed relative to the line level (0 dB), which is usually 15-20 dB below the maximum level. Compression is obtained when $R > 1$, $R = 1$ does not affect the signal and $R < 1$ gives rise to expansion.

- Maximum Amplitude: the upper boundary of the active input range, relative to the line level (0 dB). Expressed in dB.
- Threshold level: the lower boundary of the active input level, relative to the line level (0 dB).
- Attack Time: determines the response of the compressor as a function of time to a step in the input level. Expressed in ms.
- Release Time: relates to the recovery time of the gain of the compressor after a loud passage. Expressed in ms.
- Make-up Gain: set to compensate for the gain loss in the compressor. Expressed in dB.

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

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

Figure 3-14: Dynamic Range Compressor Effect Unit Icon



3.13.9 PROCESSING UNIT

The Processing Unit (PU) represents a functional block inside the Audio Function that transforms a number of logical input channels, grouped into one or more Clusters into a number of logical output channels, grouped into one 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.

If it is necessary to be able to bypass the functionality incorporated in the Processing Unit, then an explicit bypass topology using a Selector Unit shall be implemented.

If the Processing Unit is an explicit member of a Power Domain, then switching the Power Domain to any Power State other than D0 shall render the Unit non-functional and its output is undefined. Explicit bypass functionality may be required to preserve the integrity of the signal path.

3.13.9.1 UP/DOWN-MIX PROCESSING UNIT

The Up/Down-mix Processing Unit provides facilities to derive m output audio channels from n input audio channels. The algorithms and transforms applied to accomplish this are not defined by this specification and can be proprietary. The input channels are grouped into one input channel Cluster that enters the Processing Unit over a single Input Pin. Likewise, all output channels are grouped into one output channel Cluster, leaving the Processing Unit over a single Output Pin.

The Up/Down-mix Processing Unit can support multiple modes of operation. The available input audio channels are dictated by the Entity 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 **waClusterDescrID()** array. Each element of the **waClusterDescrID ()** array indicates which output channels in the output Cluster are effectively used in a particular mode. The unused output channels in the output Cluster shall produce muted output. Mode selection is implemented using the Get/Set Mode Select 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 Cluster to the Up/Down-mix Processing Unit therefore contains Left, Right, Center, Left Surround, Right Surround and LFE logical channels.

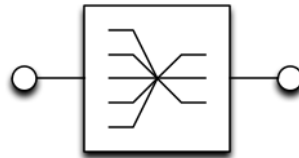
Suppose the Audio Function's hardware is limited to reproducing only dual channel audio. Then the Up/Down-mix Processing Unit could use some (sophisticated) algorithms to down-mix the available spatial audio information into two ('enriched') channels so that the maximum spatial effects can be experienced, using only two channels. It is left to the implementation to use the appropriate down-mix algorithm depending on the physical nature of the Output Terminal to which the Up/Down-mix Processing Unit is 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 example, a full-fledged MPEG-2 7.1 system). Now the Up/Down-mix Processing Unit could use certain

techniques to derive meaningful content for the extra audio channels (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 3-15: Up/Down-mix Processing Unit Icon

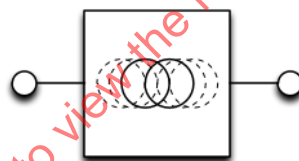


3.13.9.2 STEREO EXTENDER PROCESSING UNIT

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

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

Figure 3-16: Stereo Extender Processing Unit Icon



3.13.9.3 MULTI-FUNCTION PROCESSING UNIT

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

The following algorithms are currently supported:

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

The Multi-Function Processing Unit descriptor contains a bitmap, indicating which algorithms are supported by the Unit. The exact implementation of these algorithms and how they may interact is implementation-dependent.

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

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

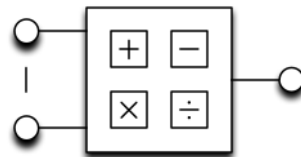
During normal operation, the Multi-Function Processing Unit transforms a number of logical input channels that enter the Unit over one or more Input Pins into a number of logical output channels that leave the Unit over a single Output Pin.

If it is necessary to be able to bypass the functionality incorporated in the Multi-Function Processing Unit, then an explicit bypass topology using a Selector Unit shall be implemented.

It is strongly recommended to implement the bypass functionality so that a generic audio driver that does not understand what functionality is implemented in the Multi-Function Processing Unit will be capable of removing it from the signal path.

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

Figure 3-17 Multi-Function Processing Unit Icon



3.13.10 EXTENSION UNIT

The Extension Unit (XU) is the method provided by this specification to easily add vendor-specific building blocks to the specification. The Extension Unit provides one or more logical input channels, grouped into one or more Clusters and transforms them into a number of logical output channels, grouped into one Cluster. Therefore, the Extension Unit can have multiple Input Pins and has a single Output Pin.

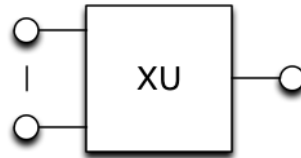
If it is necessary to be able to bypass the functionality incorporated in the Extension Unit, then an explicit bypass topology using a Selector Unit shall be implemented.

It is strongly recommended to implement the bypass functionality so that a generic audio driver that does not understand what functionality is implemented in the Extension Unit will be capable of removing it from the signal path.

If the Extension Unit is an explicit member of a Power Domain, then switching the Power Domain to any Power State other than D0 shall render the Unit non-functional and its output is undefined. Explicit bypass functionality may be required to preserve the integrity of the signal path.

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

Figure 3-18: Extension Unit Icon



3.13.11 CLOCK ENTITIES

Clock Entities are special in the sense that they do not directly manipulate logical audio streams. Instead, they provide the functionality needed to manipulate sampling clock signals and clock routing for the different Input and Output Terminals within the Audio Function. A Terminal inside the Audio Function can only be connected to one Clock Entity. The clock signal present at the Clock Input Pin of a Terminal determines the sampling frequency at which the underlying hardware is operating.

3.13.11.1 CLOCK SOURCE

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

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

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

The Clock Source Entity descriptor contains a field that indicates the origin of the actual clock signal, represented by the Entity. Furthermore, since Input and Output Terminals only have a Clock Input Pin, a clock signal can never be generated from a Terminal directly.

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

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

Figure 3-19: Clock Source Icon



3.13.11.2 CLOCK SELECTOR

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

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

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

Figure 3-20: Clock Selector Icon



3.13.11.3 CLOCK MULTIPLIER

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

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

Figure 3-21: Clock Multiplier Icon



3.14 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 Association. If the device contains multiple independent Audio Functions, there shall be multiple Audio Interface Associations, each providing full access to their associated Audio Function.

As an example of a composite device, consider a 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 is selected (Alternate Setting y). This interface also consists of a single isochronous endpoint, now receiving a data stream that interleaves Front Left and Front Right channel samples. The receiving Input Terminal now splits the stream into a Front Left and Front Right logical channel. The AudioControl interface Alternate Setting remains unchanged.

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

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

3.14.1 AUDIOCONTROL INTERFACE

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

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

The AudioControl interface is the single entry point to access the internals of the Audio Function. All requests that are concerned with the manipulation of certain Audio Controls within the Audio Function's Clock Entities, Units or Terminals shall be directed to the AudioControl interface of the Audio Function. Likewise, all descriptors related to the internals of the Audio Function are part of the class-specific AudioControl interface descriptor.

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

3.14.1.1 CONTROL ENDPOINT

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

3.14.1.2 INTERRUPT ENDPOINT

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

3.14.2 AUDIOSTREAMING INTERFACE

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

An AudioStreaming interface can have Alternate Settings that can be used to change certain characteristics of the interface and underlying endpoint. A typical use of Alternate Settings is to provide a way to change the subframe size and/or number of channels on an active AudioStreaming interface. Whenever an AudioStreaming interface requires an isochronous data endpoint, it shall at least provide the default Alternate Setting (Alternate Setting 0) with zero bandwidth requirements (no isochronous data endpoint defined) and an additional Alternate Setting that contains the actual isochronous data endpoint.

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

For every isochronous OUT or IN endpoint defined in any of the AudioStreaming interfaces, there shall be a corresponding Input or Output Terminal defined in the Audio Function. For the Host to fully understand the nature and behavior of the connection, it needs to take into account the interface- and endpoint-related descriptors as well as the Terminal-related descriptor.

3.14.2.1 ISOCHRONOUS AUDIO DATA STREAM ENDPOINT

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

Requests to control properties that exist within an Audio Function, such as volume or mute cannot be sent to the endpoint in an AudioStreaming interface. An AudioStreaming interface operates on audio data streams and

is unaware of the number of logical channels it eventually serves. Instead, these requests shall 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 shall be communicated between Host and Audio Function, they shall be clustered into one physical Cluster by interleaving the individual audio data, and the result can be directed to the single endpoint.

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

3.14.2.2 ISOCHRONOUS FEEDBACK ENDPOINT

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

3.14.2.3 AUDIO DATA FORMAT

The format used to transport audio data over the USB is entirely determined by the bits set in the **bmFormats** field of the class-specific interface descriptor. Some additional fields in this descriptor further describe the format. For details about the defined Format Types and associated data formats, see the separate document, *USB Audio Data Formats* that is considered part of this specification. Vendor-specific protocols shall be fully documented by the manufacturer.

3.14.3 CLOCK MODEL

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

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

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

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

3.14.4 POWER DOMAINS MODEL

Managed Power Domain support is optional for an Audio Function. If supported, the Audio Function is subdivided into one or more Power Domains, identified by their unique Power Domain ID, that can be

individually manipulated by the Host to optimize overall power consumption. If supported, the AudioControl interface contains an array of Power Domain Controls that allows the Host to set Power Domain States for each individual Power Domain inside the Audio Function. The Power Domain Control array has as many elements as there are Power Domains and access to a specific Power Domain Control is based on the Power Domain ID.

A Power Domain shall support three possible Power Domain States D0 to D2. Power Domain State D0 is the fully operational state and shall be the default Power Domain State for all Power Domains. Power Domain State D1 is a non-functional state that shall consume less power than Power Domain State D0, but still has the capability to generate wake-up events (interrupts) for the Controls in its Power Domain. Power Domain State D2 shall consume even less (possibly zero) power than Power Domain State D1, and shall not be able to generate any wake-up events for the Controls in its Power Domain.

An Audio Function is not allowed to change the Power Domain State of any of its Power Domains autonomously. A change in Power Domain State shall always be initiated through an explicit request from the Host. Power Domain States shall be retained across function or device suspend.

Note: Power Domain States operate independently from the function or device suspend state. For example, bringing all Power Domains in an Audio Function into Power Domain State D2 does not automatically put the Audio Function into function suspend.

Actual power consumption levels for each Power Domain State are not advertised. The only requirement is that a higher numbered Power Domain State consume less power than all lower numbered Power Domain States, potentially at the expense of higher recovery times. Recovery time is defined as the approximate time it takes to get from the lower Power Domain State D1 or D2 back to the fully operational Power Domain State D0. The Power Domain descriptor shall indicate the recovery time for the transition from Power Domain State D1 to D0 and from Power Domain State D2 to D0.

The Audio Function is best placed to manage the details of its resources and their power consumption under various conditions. These details are therefore not exposed to the Host. Rather, the Host indicates to the Audio Function which parts of the Audio Function it intends to use at the current time and the Audio Function shall autonomously decide what resources it can safely bring into a lower power state.

With a few exceptions, the Host expresses to the Audio Function how it will use its building blocks by indicating which Input and Output Terminals it will actively use to accommodate a certain usage scenario. From that information, the Audio Function can derive on its own what internal resources (Entities) can be brought into a lower power state or even shut down completely for that usage scenario. Therefore, there is no need to exhaustively enumerate membership to a certain Power Domain for all Entities in the Audio Function. It suffices to indicate Power Domain membership for Input and Output Terminals only. By manipulating a particular Power Domain's State, the Host now indicates to the Audio Function whether it wants to use the inputs and/or outputs associated with that Power Domain in the current usage scenario.

It is possible to have a Power Domain that only includes Entities other than Input and Output Terminals. In this case, an explicit bypass of these Entities should be implemented in the Audio Function's topology via a Selector Unit that resides outside that Power Domain. Setting the Power Domain State of the Power Domain to which such an Entity belongs to anything other than D0, shall render that Entity non-functional and therefore it shall cease to produce meaningful output (the output is undefined by this specification in this case). To preserve the integrity of the signal path, Host software needs to bypass the Entity in question first (by selecting the appropriate input on the Selector Unit) before switching the Power Domain to a lower Power State.

Entities that can be explicitly bypassed may be the sole member of a Power Domain or may be a member of a larger membership, provided all non-Terminal members of the membership can be bypassed either individually or as a group.

3.14.5 ADDITIONAL POWER CONSIDERATIONS AND REQUIREMENTS

An Audio 3.0 compliant bus-powered device shall support LPM/L1 according to the requirements defined in the USB 2.0 LPM/L1 ECN and LPM Errata. The Audio 3.0 device USB subsystem shall implement and enter low-power mode during L1. An Audio 3.0 device shall accept LPM/L1 entry requests (i.e. return an ACK) at all times, even if the device's isochronous RX endpoint buffer has not yet been emptied or the isochronous TX endpoint already contains samples for the next burst. The Host is responsible for and must be trusted to resume the device in time to service the next isochronous burst.

The only exception is when the device needs additional uSOFs tokens in order to resynchronize its internal clock with the USB clock. In such cases the device may deny the request to put the link into an L1 state. It shall do so only once every 64 ms or longer.

A USB Host that supports LPM/L1 is required to follow the requirements defined in the USB 2.0 LPM/L1 ECN and LPM Errata. However, for isochronous endpoints the host shall also be required to send at least one uSOF token before resuming transactions to an endpoint. The USB Host shall be required to request LPM/L1 entry after servicing the device in each service interval. The USB host is additionally required to implement low power state and save power during L1.

3.14.6 BINDING BETWEEN PHYSICAL BUTTONS AND AUDIO CONTROLS

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

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

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

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

3.14.6.1 PHYSICAL BUTTON IS A HID CONTROL

In this case, the physical button is completely separate from the Audio Function and is implemented within the device's HID interface. The Audio Function is not even aware of the button's existence. Any change of state for the button is communicated to Host software via HID reports. It is then up to Host software to interpret the button state change and derive from there the appropriate action to be taken toward the Audio Function. Therefore, the binding responsibility resides entirely within the application or Operating System software. Although this method provides extensive flexibility, it also puts the burden of providing the correct binding on the software, making it sometimes hard to create generic application or OS software that generates the proper (manufacturer intended) binding.

3.14.6.2 PHYSICAL BUTTON IS INTEGRAL PART OF THE AUDIO CONTROL

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

IECNORM.COM : Click to view the full PDF of IEC 62680-1-5:2019

4 DESCRIPTORS

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

4.1 STANDARD DESCRIPTORS

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

Because any Device compliant with this specification is required to contain multiple Configuration descriptors, it is also required that any such device include a Configuration Summary Descriptor as part of the standard BOS descriptor. These Configuration Summary descriptors may be used by Host software to decide which Configuration to use to obtain the desired functionality, not just for any Audio Function, but also for any other function available in the Configuration. The Audio Function portion of the Configuration Summary descriptor shall contain the **bFunctionClass**, **bFunctionSubClass**, and **bFunctionProtocol** fields of each AIA contained in the Configuration being summarized. As mentioned previously in Section 3.3, “Backwards Compatibility”, the same AIA may need to be included in multiple Configuration descriptors if the USB Device contains other USB Class Functions with differing versions.

Refer to the BOS Configuration Summary ECR/ECN for more details.

Also note that because Devices that are compliant with this specification are required to contain a BOS descriptor, the minimum **bDeviceProtocol** value in the Standard USB Device descriptor shall be 0x0201.

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

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

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

4.2 CLASS-SPECIFIC DESCRIPTORS

This specification allows for two different methods to express class-specific descriptors. The first method follows the traditional layout for USB descriptors whereas the second method introduces a new way for expressing and retrieving class-specific descriptors. These descriptors are called High Capability descriptors. For each class-specific descriptor, implementations shall use the method as indicated in this specification. High Capability descriptors are used in the following cases:

- The descriptor length exceeds the 256-byte limit imposed by traditional descriptors
- The descriptor is dynamic in nature as per this specification, i.e. it may change after enumeration (High Capability descriptors have the ability to report changes).

High Capability descriptors are never part of the configuration descriptor hierarchy, returned by the Get Configuration request. Only traditional layout descriptors can be included in this hierarchy. A High Capability descriptor shall always be referenced by a traditional class-specific descriptor that includes the High Capability descriptor’s unique ID as one of its fields.

4.2.1 TRADITIONAL CLASS-SPECIFIC DESCRIPTORS

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

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

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

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

Table 4-1: Traditional Class-Specific Descriptor Layout

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

4.2.1.1 COMMON FIELDS IN SOME CLASS-SPECIFIC DESCRIPTORS

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

4.2.1.1.1 BMCONTROLS FIELD

The **bmControls** field contains a set of bit pairs, indicating which Controls are present in the Entity and what their capabilities are. If a certain Control is not present, then the bit pair shall be set to 0b00. If a Control is present but Read-Only, the bit pair shall be set to 0b01. If a Control is also Host programmable, the bit pair shall be set to 0b11 (Read/Write). The value 0b10 is not allowed.

4.2.2 HIGH CAPABILITY CLASS-SPECIFIC DESCRIPTORS

This specification defines a new type of class-specific descriptors, called High Capability descriptors. A High Capability descriptor is not retrieved from the Audio Function during enumeration time. Rather, it can only be retrieved using the Get High Capability Descriptor request as defined in Section 5.2.3.3, “High Capability Descriptor Request”.

A High Capability descriptor is always referenced by another descriptor via its unique ID.

A High Capability class-specific descriptor can be up to 64K bytes in length. Also, it can generate an interrupt of source type **HIGH_CAPABILITY_DESCRIPTOR** whenever the Audio Function changes the descriptor during normal operation (dynamic descriptor).

High Capability descriptors all follow a common layout, very similar to the traditional class-specific descriptors. The first three fields of any High Capability descriptor are common to all High Capability descriptors and are followed by a layout that is specific to the type and subtype of the descriptor.

The **wLength** field contains the total length of the descriptor, in bytes. Descriptor lengths up to a maximum of 64K bytes are supported.

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

The **bDescriptorSubtype** field further qualifies the exact nature of the descriptor. **bDescriptorSubtype** field bits D6..0 are used to encode the descriptor subtype.

The **wDescriptorID** field contains a value that uniquely identifies the High Capability descriptor within an interface or endpoint.

Table 4-2: High Capability Class-Specific Descriptor Layout

Offset	Field	Size	Value	Description
0	wLength	2	Number	Size of this descriptor, in bytes: 6+n.
2	bDescriptorType	1	0b001xxxxx	Descriptor type.
3	bDescriptorSubtype	1	Constant	Descriptor subtype.
4	wDescriptorID	2	Number	Unique ID for this High Capability class-specific descriptor.
6	---	n	---	Descriptor type- and subtype-specific descriptor layout.

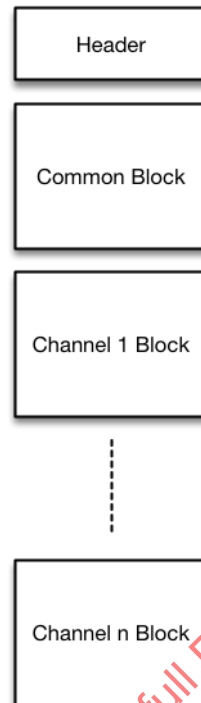
4.3 CLUSTER DESCRIPTOR

A Cluster is a grouping of audio channels that share the same characteristics like sampling frequency, bit resolution, etc. To characterize a Cluster, a Cluster descriptor is introduced.

This specification introduces a Cluster descriptor that is significantly different from earlier Cluster descriptor definitions. The goal is to be much more flexible and extensible and at the same time describe the Cluster more accurately and with richer content.

The Cluster descriptor always uses the High Capability representation. At the highest level, it is structured as follows:

Figure 4-1: Cluster Descriptor



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

4.3.1 CLUSTER DESCRIPTOR HEADER

The Header starts with the **wLength** field that contains the total number of bytes in the entire descriptor.

The **wDescriptorID** field contains an ID number that uniquely identifies the descriptor within the Audio Function. The value zero is reserved and shall not be used as a valid Cluster descriptor ID.

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

Table 4-3: Cluster Descriptor Header

Offset	Field	Size	Value	Description
0	wLength	2	Number	Total length of the Cluster descriptor, in bytes.
2	bDescriptorType	1	Constant	CS_CLUSTER descriptor type.
3	bDescriptorSubtype	1	Constant	SUBTYPE_UNDEFINED descriptor subtype.
4	wDescriptorID	2	Number	Unique ID of this Cluster descriptor.
6	bNrChannels	1	Number	Number of channels present in the Cluster: n.

4.3.2 CLUSTER DESCRIPTOR BLOCK

The Cluster descriptor Header is followed by a number of Cluster descriptor Blocks. There is an optional Common Block, followed by as many Channel Blocks as there are channels in the Cluster. Each Block consists of a number of Segments.

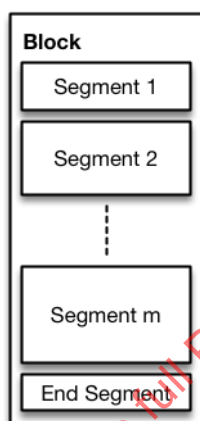
The Common Block consists of Segments that contain relevant information about characteristics of the Cluster as a whole.

A Channel Block consists of Segments that contain relevant information about that channel's characteristics. At a minimum, each Channel Block shall contain either an Information Segment or an Ambisonic Segment and a single End Segment. All other Segments are optional. It is highly recommended that the same layout is used for each Channel Block, i.e. the same Segments appear in the same order in each Channel Block.

However, if for any reason, there is currently no Cluster information available, then the Cluster descriptor shall only contain a Header with the **bNrChannels** field set to zero. This is called the Empty Cluster descriptor.

Figure 4-2 further illustrates the above concepts.

Figure 4-2: Cluster Descriptor Block



4.3.2.1 SEGMENTS

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

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

The layout for a Segment is always as follows:

Table 4-4: Cluster Descriptor Segment

Offset	Field	Size	Value	Description
0	wLength	2	Number	Length of the Segment, in bytes: 3+n.
2	bSegmentType	1	Constant	Describes the Segment Type and the type of content in the Segment.
3	Segment-specific	n		Segment-specific content.

4.3.2.1.1 END SEGMENT

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

Table 4-5: End Segment

Offset	Field	Size	Value	Description
0	wLength	2	Number	Length of the End Segment, in bytes: 3.
2	bSegmentType	1	Constant	END_SEGMENT.

4.3.2.1.2 COMMON BLOCK SEGMENTS

The following Common Block Segment types are defined:

- Cluster Description
- Vendor-specific

Values for the Common Block Segment types can be found in Appendix A.10, “Cluster Descriptor Segment types.”

4.3.2.1.2.1 CLUSTER DESCRIPTION SEGMENT

The Cluster Description Segment contains the ID of a String descriptor in the **wCDDescrStr** field that provides more information about the Cluster.

Table 4-6: Cluster Description Segment

Offset	Field	Size	Value	Description
0	wLength	2	Number	Length of the Segment, in bytes: 5.
2	bSegmentType	1	Constant	CLUSTER_DESCRIPTION.
3	wCDDescrStr	2	wStrDescrID	ID of a String descriptor that further describes the Cluster.

4.3.2.1.2.2 VENDOR-DEFINED SEGMENT

Vendors are allowed to add vendor-defined Segments to the Common Block to convey additional, proprietary Cluster information. The vendor-defined Segment shall use the following layout:

Table 4-7: Vendor-defined Segment

Offset	Field	Size	Value	Description
0	wLength	2	Number	Length of the End Segment, in bytes: 3+n.
2	bSegmentType	1	Constant	CLUSTER_VENDOR_DEFINED.
3	Vendor-defined	n	N/A	Vendor-defined extension of the Segment.

4.3.2.1.3 CHANNEL BLOCK SEGMENTS

The following Channel Block Segment types are defined:

- Information
- Ambisonic
- Channel Description
- Vendor-specific

Values for the Channel Block Segment types can be found in Appendix A.10, “Cluster Descriptor Segment types.”

4.3.2.1.3.1 INFORMATION SEGMENT

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

The **bChPurpose** field indicates the purpose of the channel. Currently defined purposes for a channel are:

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

The values for the purposes listed above can be found in Appendix A.11, “Channel Purpose Definitions.”

Note that great care should be taken when indicating purposes for an audio channel. Only when the channel has been specifically tailored to serve a purpose should the appropriate bit(s) be set. For example, if raw audio is processed specifically to be used by applications that use human voice as input, such a channel may be marked Voice.

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

Table 4-8: Channel Relationships

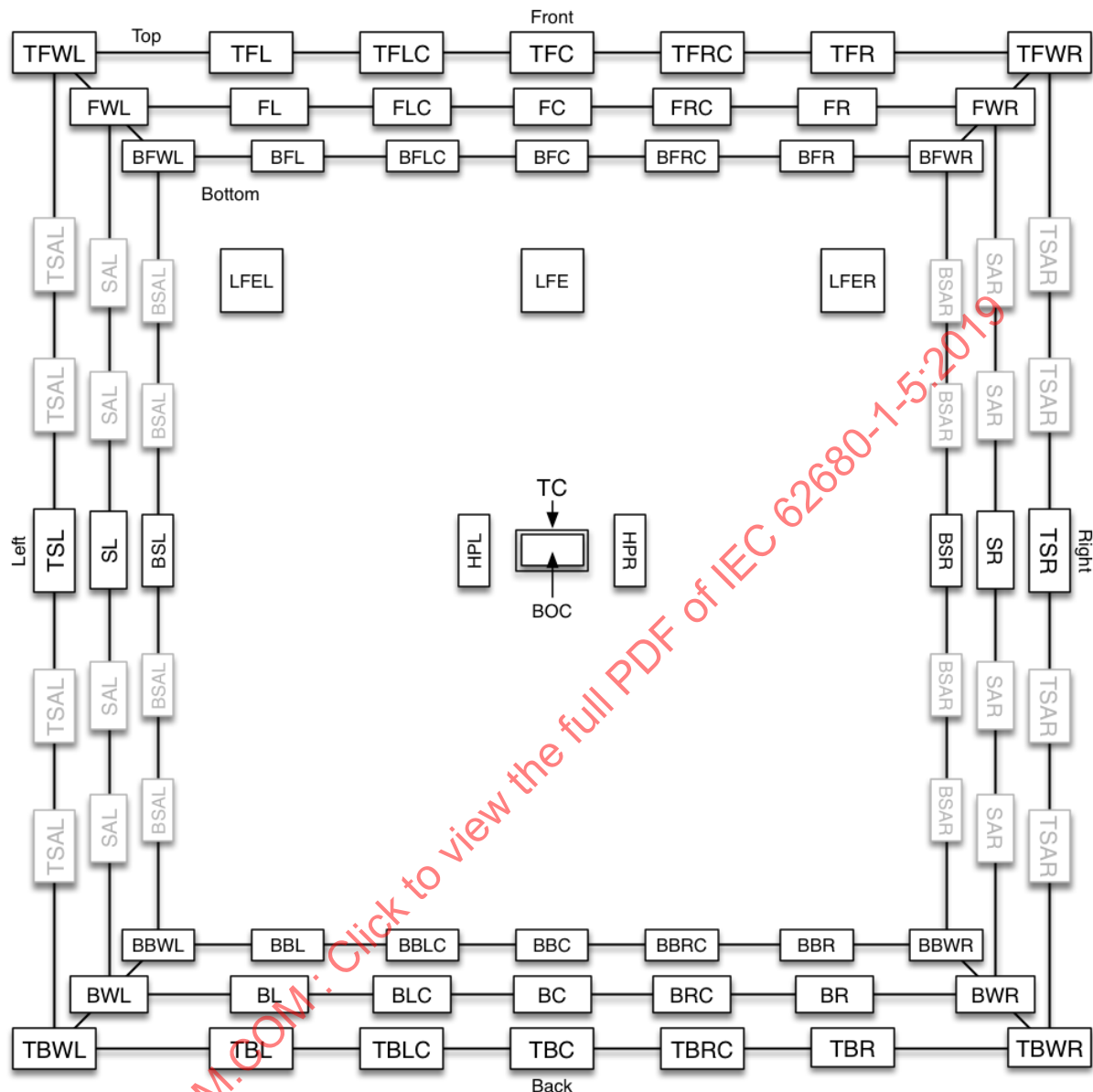
	Channel Relationship	Acronym	Description
Generic	RELATIONSHIP_UNDEFINED	UND	Relationship undefined or unknown
	MONO	M	A mono channel
	LEFT	L	A generic Left channel
	RIGHT	R	A generic Right channel
	ARRAY	AR	A channel that is part of an array configuration
Input-Related	PATTERN_X	PX	The X channel of an X/Y microphone
	PATTERN_Y	PY	The Y channel of an X/Y microphone
	PATTERN_A	PA	The A channel of an A/B microphone
	PATTERN_B	PB	The B channel of an A/B microphone
	PATTERN_M	PM	The M channel of an M/S microphone
	PATTERN_S	PS	The S channel of an M/S microphone
Output-Related	Front Left	FL	Refer to Figure 4-3
	Front Right	FR	Refer to Figure 4-3
	Front Center	FC	Refer to Figure 4-3
	Front Left of Center	FLC	Refer to Figure 4-3
	Front Right of Center	FRC	Refer to Figure 4-3
	Front Wide Left	FWL	Refer to Figure 4-3
	Front Wide Right	FWR	Refer to Figure 4-3

	Channel Relationship	Acronym	Description
	Side Left	SL	Refer to Figure 4-3
	Side Right	SR	Refer to Figure 4-3
	Surround Array Left	SAL	Refer to Figure 4-3
	Surround Array Right	SAR	Refer to Figure 4-3
	Back Left	BL	Refer to Figure 4-3
	Back Right	BR	Refer to Figure 4-3
	Back Center	BC	Refer to Figure 4-3
	Back Left of Center	BLC	Refer to Figure 4-3
	Back Right of Center	BRC	Refer to Figure 4-3
	Back Wide Left	BWL	Refer to Figure 4-3
	Back Wide Right	BWR	Refer to Figure 4-3
	Top Center	TC	Refer to Figure 4-3
	Top Front Left	TFL	Refer to Figure 4-3
	Top Front Right	TFR	Refer to Figure 4-3
	Top Front Center	TFC	Refer to Figure 4-3
	Top Front Left of Center	TFLC	Refer to Figure 4-3
	Top Front Right of Center	TFRC	Refer to Figure 4-3
	Top Front Wide Left	TFWL	Refer to Figure 4-3
	Top Front Wide Right	TFWR	Refer to Figure 4-3
	Top Side Left	TSL	Refer to Figure 4-3
	Top Side Right	TSR	Refer to Figure 4-3
	Top Surround Array Left	TSAL	Refer to Figure 4-3
	Top Surround Array Right	TSAR	Refer to Figure 4-3
	Top Back Left	TBL	Refer to Figure 4-3
	Top Back Right	TBR	Refer to Figure 4-3
	Top Back Center	TBC	Refer to Figure 4-3
	Top Back Left Of Center	TBLC	Refer to Figure 4-3
	Top Back Right Of Center	TBRC	Refer to Figure 4-3
	Top Back Wide Left	TBWL	Refer to Figure 4-3
	Top Back Wide Right	TBWR	Refer to Figure 4-3
	Bottom Center	BC	Refer to Figure 4-3
	Bottom Front Left	BFL	Refer to Figure 4-3
	Bottom Front Right	BFR	Refer to Figure 4-3
	Bottom Front Center	BFC	Refer to Figure 4-3
	Bottom Front Left Of Center	BFLC	Refer to Figure 4-3
	Bottom Front Right Of Center	BFRC	Refer to Figure 4-3
	Bottom Front Wide Left	BFWL	Refer to Figure 4-3
	Bottom Front Wide Right	BFWR	Refer to Figure 4-3
	Bottom Side Left	BSL	Refer to Figure 4-3

	Channel Relationship	Acronym	Description
	Bottom Side Right	BSR	Refer to Figure 4-3
	Bottom Surround Array Left	BSAL	Refer to Figure 4-3
	Bottom Surround Array Right	BSAR	Refer to Figure 4-3
	Bottom Back Left	BBL	Refer to Figure 4-3
	Bottom Back Right	BBR	Refer to Figure 4-3
	Bottom Back Center	BBC	Refer to Figure 4-3
	Bottom Back Left Of Center	BBLC	Refer to Figure 4-3
	Bottom Back Right Of Center	BBRC	Refer to Figure 4-3
	Bottom Back Wide Left	BBWL	Refer to Figure 4-3
	Bottom Back Wide Right	BBWR	Refer to Figure 4-3
	Low Frequency Effects	LFE	Refer to Figure 4-3
	Low Frequency Effects Left	LFEL	Refer to Figure 4-3
	Low Frequency Effects Right	LFER	Refer to Figure 4-3
	Headphone Left	HPL	Refer to Figure 4-3
	Headphone Right	HPR	Refer to Figure 4-3

IECNORM.COM : Click to view the full PDF of IEC 62680-1-5:2019

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



Constant names, acronyms, and their associated values are listed in Appendix A.12, “Channel Relationship Definitions.”

The **bChGroupID** field is used to create a link among a subset of channels in the Cluster by specifying the same value in the **bChGroupID** field for those channels. This field is useful when multiple sets of related channels are present in the same Cluster. For example, a single Cluster may carry a Group of 5.1 channels and another Group of microphone channels. The 5.1 channels would share the same Group ID value, indicating that they belong to a (5.1) Group, and the microphone channels would share another Group ID value, indicating they belong to a different (microphone) Group. The scope of the **bChGroupID** field is limited to the Cluster to which the channels belong. Note that specifying a value of 0 in this field for all channels in the Cluster effectively creates a Group with ID=0 to which all channels belong.

Table 4-9: Information Segment

Offset	Field	Size	Value	Description
0	wLength	2	Number	Length of the Channel Information Segment, in bytes: 6.
2	bSegmentType	1	Constant	CHANNEL_INFORMATION.
3	bChPurpose	1	Number	Intended purpose of the channel.
4	bChRelationship	1	Number	Describes the relationship of this channel with the other channels. Refer to Appendix A.12, “Channel Relationship Definitions.”
5	bChGroupID	1	Number	ID used to group channels together.

4.3.2.1.3.2 AMBISONIC SEGMENT

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

The **bCompOrdering** field indicates the convention used for ordering of the spherical harmonics. See Appendix A.13, “Ambisonic Component Ordering Convention Types” for the supported component ordering conventions. All channels in a Group (see below) shall indicate the same component ordering convention.

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

The **bAmbNorm** field indicates the type of normalization used for the channel. See Appendix A.14, “Ambisonic Normalization Types” for the supported normalization types. All channels in a Group (see below) shall indicate the same normalization type.

The **bChGroupID** field is used to create a link among a subset of channels in the Cluster by specifying the same value in the **bChGroupID** field for those channels. This field is useful when multiple sets of related channels are present in the same Cluster. For example, a single Cluster may carry a Group of 5.1 channels and another Group of microphone channels. The 5.1 channels would share the same Group ID value, indicating that they belong to a (5.1) Group, and the microphone channels would share another Group ID value, indicating they belong to a different (microphone) Group. The scope of the **bChGroupID** field is limited to the Cluster to which the channels belong. Note that specifying a value of 0 in this field for all channels in the Cluster effectively creates a Group with ID=0 to which all channels belong.

Table 4-10: Ambisonic Segment

Offset	Field	Size	Value	Description
0	wLength	2	Number	Length of the Ambisonic Channel Information Segment, in bytes: 7.
2	bSegmentType	1	Constant	CHANNEL_AMBISONIC.
3	bCompOrdering	1	Number	The Component Ordering convention for the Ambisonic Spherical Harmonics. Refer to Appendix A.13, “Ambisonic Component Ordering Convention Types.”
4	bACN	1	Number	Ambisonic Channel Number.
5	bAmbNorm	1	Number	Applied channel normalization. Refer to Appendix A.14, “Ambisonic Normalization Types.”
6	bChGroupID	1	Number	ID used to group channels together.

4.3.2.1.3.3 CHANNEL DESCRIPTION SEGMENT

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

Table 4-11: Channel Description Segment

Offset	Field	Size	Value	Description
0	wLength	2	Number	Length of the Segment, in bytes: 5.
2	bSegmentType	1	Constant	CHANNEL_DESCRIPTION.
3	wChDescrStr	2	wStrDescrID	ID of a String descriptor that further describes the channel.

4.3.2.1.3.4 VENDOR-DEFINED SEGMENT

Vendors are allowed to add vendor-defined Segments to the Channel Block to convey additional, proprietary channel information. The vendor-defined Segment shall use the following layout:

Table 4-12: Vendor-defined Segment

Offset	Field	Size	Value	Description
0	wLength	2	Number	Length of the End Segment, in bytes: 3+n.
2	bSegmentType	1	Constant	CHANNEL_VENDOR_DEFINED.
3	Vendor-defined	n	N/A	Vendor-defined extension of the Segment.

4.3.3 EXAMPLE CLUSTER DESCRIPTOR

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

Table 4-13: Cluster Descriptor Example

	Offset	Field	Size	Value	Description
Header	0	wLength	2	0x3E	Total length of the Cluster descriptor, in bytes: 62.
	2	bDescriptorType	1	0x26	CS_CLUSTER descriptor type.
	3	bDescriptorSubtype	1	0x00	SUBTYPE_UNDEFINED descriptor subtype.
	4	wDescriptorID	2	Implementation-dependent	Unique ID of this Cluster descriptor.
	6	bNrChannels	1	6	Number of channels present in the Cluster.
Information Segment	8	wLength	2	0x06	Length of the Information Segment, in bytes.
	10	bSegmentType	1	0x20	CHANNEL_INFORMATION.
	11	bChPurpose	1	0x00	Generic Audio.
	12	bChRelationship	1	0x80	Front Left.
	13	bChGroupID	1	0x01	ID used to group channels together.
End	14	wLength	2	0x03	Length of the End Segment, in bytes.
	16	bSegmentType	1	0xFF	END_SEGMENT.
Information	17	wLength	2	0x06	Length of the Information Segment, in bytes.
	19	bSegmentType	1	0x20	CHANNEL_INFORMATION.
	20	bChPurpose	1	0x00	Generic Audio.
	21	bChRelationship	1	0x81	Front Right.

	Offset	Field	Size	Value	Description
End	22	bChGroupID	1	0x01	ID used to group channels together.
	23	wLength	2	0x03	Length of the End Segment, in bytes.
	25	bSegmentType	1	0xFF	END_SEGMENT.
Information	26	wLength	2	0x06	Length of the Information Segment.
	28	bSegmentType	1	0x20	CHANNEL_INFORMATION.
	29	bChPurpose	1	0x00	Generic Audio.
	30	bChRelationship	1	0x82	Front Center.
	31	bChGroupID	1	0x01	ID used to group channels together.
End	32	wLength	2	0x03	Length of the End Segment, in bytes.
	34	bSegmentType	1	0xFF	END_SEGMENT.
Information	35	wLength	2	0x06	Length of the Information Segment.
	37	bSegmentType	1	0x20	CHANNEL_INFORMATION.
	38	bChPurpose	1	0x00	Generic Audio.
	39	bChRelationship	1	0x89	Surround Array Left.
	40	bChGroupID	1	0x01	ID used to group channels together.
End	41	wLength	2	0x03	Length of the End Segment.
	43	bSegmentType	1	0xFF	END_SEGMENT.
Information	44	wLength	2	0x06	Length of the Information Segment.
	46	bSegmentType	1	0x20	CHANNEL_INFORMATION.
	47	bChPurpose	1	0x00	Generic Audio.
	48	bChRelationship	1	0x8A	Surround Array Right.
	49	bChGroupID	1	0x01	ID used to group channels together.
End	50	wLength	2	0x03	Length of the End Segment.
	52	bSegmentType	1	0xFF	END_SEGMENT.
Information	53	wLength	2	0x06	Length of the Information Segment.
	55	bSegmentType	1	0x20	CHANNEL_INFORMATION.
	56	bChPurpose	1	0x00	Generic Audio.
	57	bChRelationship	1	0xB8	Low Frequency Effects.
	58	bChGroupID	1	0x01	ID used to group channels together.
End	59	wLength	2	0x03	Length of the End Segment.
	61	bSegmentType	1	0xFF	END_SEGMENT.

4.3.4 CEA-861.2 CHANNEL MAPPING

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

The mapping between the USB-defined channel relationships and the CEA speaker allocations is included in the table in Appendix A.12, “Channel Relationship Definitions.”

4.4 PHYSICAL VERSUS LOGICAL CLUSTER

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

- Logical Cluster descriptor
- Physical Cluster descriptor

The layout of these descriptors is identical and shall always use the High Capability representation as described in Section 4.2.2, “High Capability Class-Specific Descriptors.” Both the logical and physical Cluster descriptors are not independent descriptors as such. They are always referenced by other descriptors. The referencing descriptors always include a **wClusterDescrID** field that contains the unique ID of the Cluster descriptor they reference.

The logical Cluster descriptor is referenced by one of the following descriptors:

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

The physical Cluster descriptor is referenced by the class-specific AudioStreaming interface descriptor in each Alternate Setting of an AudioStreaming interface (except for Alternate Setting 0).

The physical Cluster descriptor is also referenced by the Connectors descriptor to provide information about the physical channels that travel over the various connectors, associated with a Terminal.

4.4.1 MAPPING BETWEEN PHYSICAL AND LOGICAL CLUSTERS

Audio streams always enter or leave the Audio Function via an AudioStreaming – Terminal pair. The following sections describe in more detail the possible cases and the internal operations that take place.

4.4.1.1 AUDIOSTREAMING OUT INTERFACE – INPUT TERMINAL

For Type I Audio streams, the physical Cluster descriptor referenced by the class-specific AudioStreaming interface descriptor of the active Alternate Setting of that interface describes the audio channels as they enter the interface.

For Type III/IV Audio streams, the physical Cluster describes the audio channels after they have been decoded into discrete audio channels.

The audio samples (Audio Subslots) shall be ordered in the exact same order as they are listed in the physical Cluster descriptor. The Cluster then enters an implicit Up/Down-mix process to produce an output Cluster as described by the Cluster descriptor that is referenced by the associated Input Terminal descriptor.

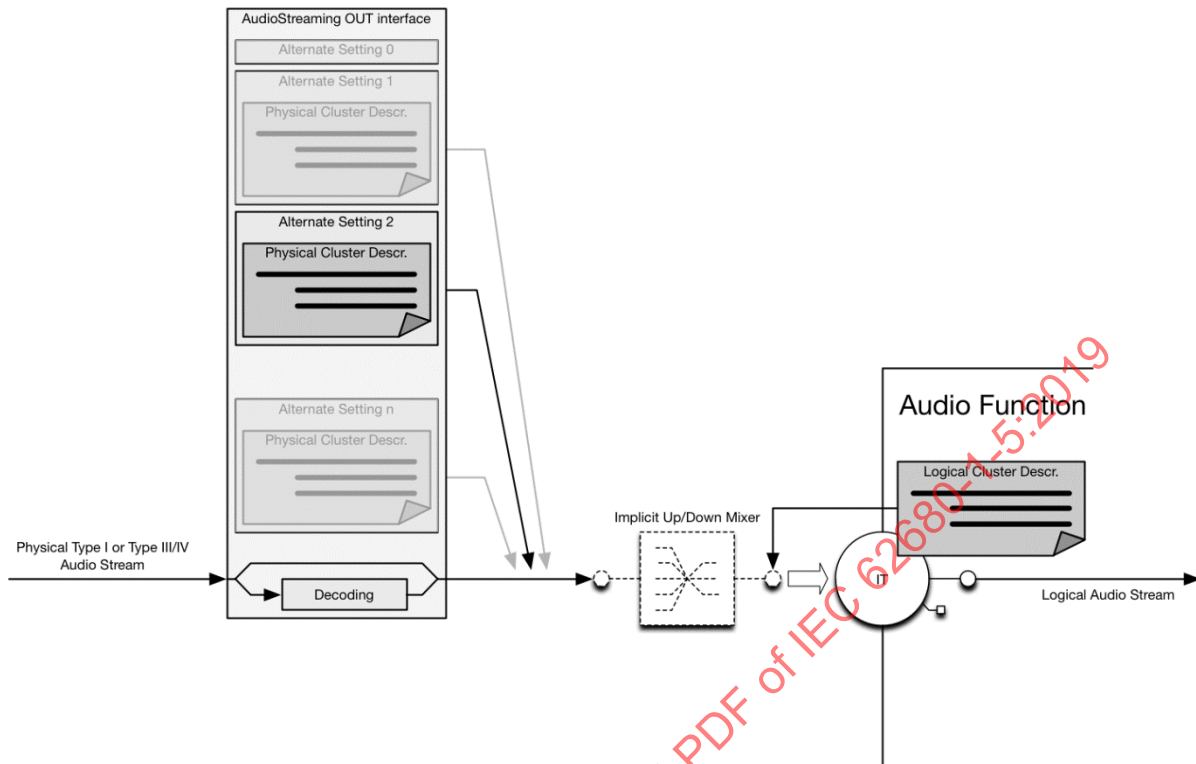
Finally, that Cluster then enters the Audio Function via the Output Pin of the Input Terminal.

This construct allows for a complete decoupling between the physical Cluster configuration as selected by the active Alternate Setting of the interface, and the logical Cluster configuration as described by the Input Terminal’s Cluster descriptor. In other words, the logical Cluster configuration is independent of the chosen Alternate Setting of the AudioStreaming interface.

The process used to perform the Up/Down-mix operation is implementation-specific and not specified here.

Figure 4-4 illustrates the concept for AudioStreaming OUT interfaces.

Figure 4-4: Physical to Logical Cluster Mapping



4.4.1.2 AUDIOSTREAMING IN INTERFACE – OUTPUT TERMINAL

For Type I Audio streams, the physical Cluster descriptor referenced by the class-specific AudioStreaming interface descriptor of the active Alternate Setting of that AudioStreaming OUT interface describes the audio channels as they enter the interface.

For Type III/IV Audio streams, the physical Cluster describes the audio channels after they have been decoded into discrete audio channels.

An upstream Cluster descriptor describes the logical Cluster entering the Input Pin of the Output Terminal.

The Cluster then enters an implicit Up/Down-mix process to produce an output Cluster as described by the physical Cluster descriptor referenced by the class-specific AudioStreaming interface descriptor of the active Alternate Setting of the interface. The audio samples (Audio Subslots) shall be ordered in the exact same order as they are listed in the physical Cluster descriptor.

For Type I Audio streams, that Cluster then leaves the AudioStreaming interface.

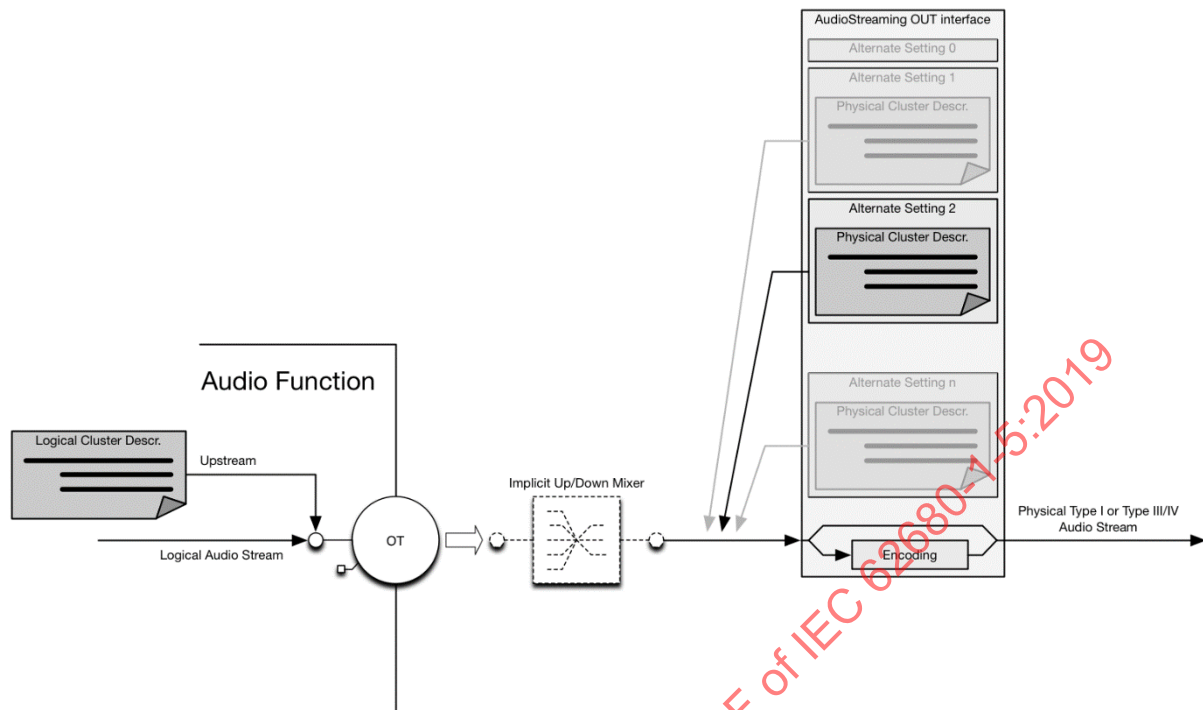
For Type III/IV Audio streams, the Cluster is encoded before leaving the AudioStreaming interface.

This construct allows for a complete decoupling between the logical Cluster configuration as described by the upstream Cluster descriptor and the physical Cluster configuration as selected by the active Alternate Setting of the interface. In other words, the physical Cluster configuration is independent of the upstream Cluster configuration and solely determined by the chosen Alternate Setting of the AudioStreaming interface.

The process used to perform the Up/Down-mix operation is implementation-specific and not specified here.

Figure 4-5 illustrates the concept for AudioStreaming IN interfaces.

Figure 4-5: Logical to Physical Cluster Mapping



4.5 AUDIOCONTROL INTERFACE DESCRIPTORS

The AudioControl (AC) interface descriptors contain all relevant information to fully characterize the corresponding Audio Function. The standard interface descriptor characterizes the interface itself, whereas the class-specific interface 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.

4.5.1 STANDARD AC INTERFACE DESCRIPTOR

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

Table 4-14: 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. Shall be set to 0.
4	bNumEndpoints	1	Number	Number of endpoints used by this interface (excluding endpoint 0). This number is either 0 or 1 if the optional interrupt endpoint is present.

Offset	Field	Size	Value	Description
5	bInterfaceClass	1	Class	AUDIO. Audio Interface Class code (assigned by the USB). See Appendix A.4, “Audio Interface Class Code.”
6	bInterfaceSubClass	1	Subclass	AUDIOCONTROL. Audio Interface Subclass code. Assigned by this specification. See Appendix A.5, “Audio Interface Subclass Codes.”
7	bInterfaceProtocol	1	Protocol	IP_VERSION_03_00 Interface Protocol code. Indicates the current version of the specification. See Appendix A.6, “Audio Interface Protocol Codes”
8	iInterface	1	Index	Index of a String descriptor that describes this interface.

4.5.2 CLASS-SPECIFIC AC INTERFACE DESCRIPTOR

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

The total length of the class-specific AC interface descriptor depends on the number of Clock Entities, Units, Terminals, and Power Domains 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 **bCategory** field contains a constant that indicates what the primary use of this Audio Function is as intended by the manufacturer.

The order in which the Clock Entity, Unit, Terminal, and Power Domain descriptors are reported is not important because every descriptor can be identified through its **bDescriptorType** and **bDescriptorSubtype** field.

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

Table 4-15: Class-Specific AC Interface Header Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	Number	Size of this descriptor, in bytes: 10.
1	bDescriptorType	1	Constant	CS_INTERFACE descriptor type.
2	bDescriptorSubtype	1	Constant	HEADER descriptor subtype.
3	bCategory	1	Constant	Constant, indicating the primary use of this Audio Function, as intended by the manufacturer. See Appendix A.7, “Audio Function Category Codes.”
4	wTotalLength	2	Number	Total number of bytes returned for the class-specific AudioControl interface descriptor. Includes the combined length of this descriptor header and all Clock Source, Unit, Terminal, and Power Domain descriptors.
6	bmControls	4	Bitmap	D1..0: Latency Control. D31..2: Reserved.

The Clock Entity descriptors, Unit descriptors, Terminal descriptors, and Power Domain descriptors appear next, in any order. The layout of the Entity descriptors depends on the type of Clock Entity, Unit, or Terminal they represent. The Power Domain descriptor has a fixed layout. There are three types of Clock Entity

descriptors, seven types of Unit descriptors, two types of Terminal descriptors, and a single type of Power Domain descriptor. The Entity descriptors are summarized in the following sections.

The first three fields are common for all descriptors. They contain the Descriptor Length, Descriptor Type, and Descriptor Subtype. Entity descriptors have an additional common field that contains the Clock Entity ID, Unit ID, Terminal ID, or Power Domain ID for the Entity.

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

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

4.5.2.1 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 AudioControl interface may have the same ID. This value shall be passed in the Entity ID field (part of the **wIndex** field) of each request that is directed to the Terminal.

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

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

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

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

The **wClusterDescrID** field contains the unique ID of the Cluster descriptor that characterizes the logical Cluster that leaves the Input Terminal over the single Output Pin ('downstream' connection). For a detailed description of the Cluster descriptor, see Section 4.3, "Cluster Descriptor".

The **wExTerminalDescrID** field contains the unique ID of the Extended Terminal descriptor that is associated with this Terminal. For a detailed description of the Extended Terminal descriptor, see Section 4.5.2.3, "Extended Terminal Descriptor."

The **wConnectorsDescrID** field contains the unique ID of the Connectors descriptor that is associated with this Terminal. For a detailed description of the Connectors descriptor, see Section 4.5.2.4, "Connectors Descriptor."

An ID of a String descriptor is provided to further describe the Input Terminal.

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

Table 4-16: Input Terminal Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	Number	Size of this descriptor, in bytes: 20.
1	bDescriptorType	1	Constant	CS_INTERFACE descriptor type.
2	bDescriptorSubtype	1	Constant	INPUT_TERMINAL descriptor subtype.
3	bTerminalID	1	Number	Value 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 <i>USB Audio Terminal Types</i> .
6	bAssocTerminal	1	Number	ID of the Output Terminal to which this Input Terminal is associated.
7	bCSourceID	1	Number	ID of the Clock Entity to which this Input Terminal is connected.
8	bmControls	4	Bitmap	D1..0: Insertion Control. D3..2: Overload Control. D5..4: Underflow Control. D7..6: Overflow Control. D31..8: Reserved.
12	wClusterDescrID	2	Number	ID of the Cluster descriptor for this Input Terminal.
14	wExTerminalDescrID	2	Number	ID of the Extended Terminal descriptor for this Input Terminal. Shall be set to zero if no Extended Terminal descriptor is present.
16	wConnectorsDescrID	2	Number	ID of the Connectors descriptor for this Input Terminal. Shall be set to zero if no Connectors descriptor is present.
18	wTerminalDescrStr	2	wStrDescrID	ID of a class-specific String descriptor, describing the Input Terminal.

4.5.2.2 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 AudioControl interface may have the same ID. This value shall be passed in the Entity ID field (part of the **wIndex** field) of each request that is directed to the Terminal.

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

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

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

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

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

The **wExTerminalDescrID** field contains the unique ID of the Extended Terminal descriptor that is associated with this Terminal. For a detailed description of the Extended Terminal descriptor, see Section 4.5.2.3, “Extended Terminal Descriptor.”

The **wConnectorsDescrID** field contains the unique ID of the Connectors descriptor that is associated with this Terminal. For a detailed description of the Connectors descriptor, see Section 4.5.2.4, “Connectors Descriptor.”

An ID of a String descriptor is provided to further describe the Output Terminal.

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

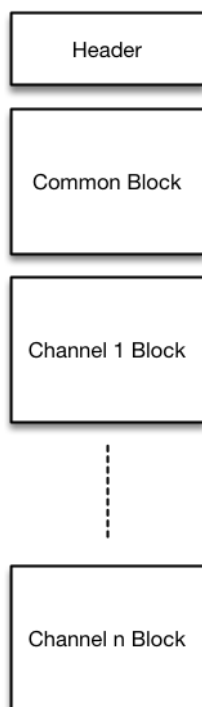
Table 4-17: Output Terminal Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	Number	Size of this descriptor, in bytes: 19.
1	bDescriptorType	1	Constant	CS_INTERFACE descriptor type.
2	bDescriptorSubtype	1	Constant	OUTPUT_TERMINAL descriptor subtype.
3	bTerminalID	1	Number	Value 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 <i>USB Audio Terminal Types</i> .
6	bAssocTerminal	1	Number	Value identifying the Input Terminal to which this Output Terminal is associated.
7	bSourceID	1	Number	ID of the Unit or Terminal to which this Terminal is connected.
8	bCSourceID	1	Number	ID of the Clock Entity to which this Output Terminal is connected.
9	bmControls	4	Bitmap	D1..0: Insertion Control. D3..2: Overload Control. D5..4: Underflow Control. D7..6: Overflow Control. D31..8: Reserved.
13	wExTerminalDescrID	2	Number	ID of the Extended Terminal descriptor for this Output Terminal. Shall be set to zero if no Extended Terminal descriptor is present.
15	wConnectorsDescrID	2	Number	ID of the Connectors descriptor for this Output Terminal. Shall be set to zero if no Connectors descriptor is present.
17	wTerminalDescrStr	2	wStrDescrID	ID of a class-specific String descriptor, describing the Output Terminal.

4.5.2.3 EXTENDED TERMINAL DESCRIPTOR

The Extended Terminal descriptor optionally returns additional physical information about the channels that enter or leave the Terminal. It always uses the High Capability representation.

Figure 4-6: Extended Terminal Descriptor



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

4.5.2.3.1 EXTENDED TERMINAL DESCRIPTOR HEADER

The Header starts with the **wLength** field that contains the total number of bytes in the entire descriptor.

The **wDescriptorID** field contains an ID number that uniquely identifies the descriptor within the Audio Function. The value zero is reserved and shall not be used as a valid Extended Terminal descriptor ID.

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

Table 4-18: Extended Terminal Descriptor Header

Offset	Field	Size	Value	Description
0	wLength	2	Number	Total length of the Cluster descriptor, in bytes.
2	bDescriptorType	1	Constant	CS_INTERFACE descriptor type.
3	bDescriptorSubtype	1	Constant	EXTENDED_TERMINAL descriptor subtype.
4	wDescriptorID	2	Number	Unique ID of this Extended Terminal descriptor.
6	bNrChannels	1	Number	Number of channels present in the Cluster of the Terminal: n.

4.5.2.3.2 EXTENDED TERMINAL DESCRIPTOR BLOCK

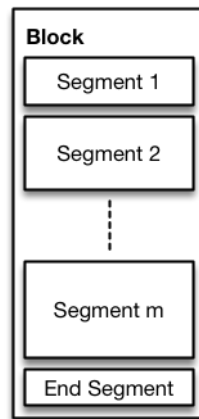
The Extended Terminal descriptor Header is followed by a number of Extended Terminal descriptor Blocks.

There is an optional Common Block, followed by as many Channel Blocks as there are channels in the Cluster of the Terminal. Each Block consists of a number of Segments.

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

A Channel Block consists of Segments that contain relevant information about that channel's physical characteristics. All Segments are optional. It is highly recommended that the same layout is used for each Channel Block, i.e. the same Segments appear in the same order in each Channel Block. Figure 4-7 further illustrates the above concepts.

Figure 4-7: Extended Terminal Channel Block



4.5.2.3.3 SEGMENTS

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

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

The layout for a Segment is always as follows:

Table 4-19: Cluster Descriptor Segment

Offset	Field	Size	Value	Description
0	wLength	2	Number	Length of the Segment, in bytes: 3+n.
2	bSegmentType	1	Constant	Describes the Segment Type and the type of content in the Segment.
3	Segment-specific	n		Segment-specific content.

4.5.2.3.3.1 END SEGMENT

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

Table 4-20: End Segment

Offset	Field	Size	Value	Description
0	wLength	2	Number	Length of the End Segment, in bytes: 3.
2	bSegmentType	1	Constant	END_SEGMENT.

4.5.2.3.3.2 COMMON BLOCK SEGMENTS

The following Common Block Segment types are defined:

- Vendor-specific

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

4.5.2.3.3.2.1 VENDOR-DEFINED SEGMENT

Vendors are allowed to add vendor-defined Segments to the Common Block to convey additional, proprietary Terminal information. The vendor-defined Segment shall use the following layout:

Table 4-21: Vendor-defined Segment

Offset	Field	Size	Value	Description
0	wLength	2	Number	Length of the End Segment, in bytes: 3+n.
2	bSegmentType	1	Constant	TERMINAL_VENDOR_DEFINED.
3	Vendor-defined	n	N/A	Vendor-defined extension of the Segment.

4.5.2.3.3.3 CHANNEL BLOCK SEGMENTS

The following Channel Block Segment types are defined:

- Bandwidth
- Magnitude Response
- Magnitude/Phase Response
- Position
- Vendor-specific

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

4.5.2.3.3.3.1 BANDWIDTH SEGMENT

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

Table 4-22: Bandwidth Segment

Offset	Field	Size	Value	Description
0	wLength	2	Number	Length of the Bandwidth Segment, in bytes: 11.
2	bSegmentType	1	Constant	CHANNEL_BANDWIDTH.
3	dMinFreq	4	Number	Lower -3 dB frequency point.
7	dMaxFreq	4	Number	Upper -3 dB frequency point.

4.5.2.3.3.3.2 MAGNITUDE RESPONSE

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

(0x7FFF) down to -127.9961 dB (0x8001) in steps of 1/256 dB or 0.00390625 dB (0x0001). In addition, code 0x8000, representing silence (i.e., $-\infty$ dB), may be used as well.

Table 4-23: Magnitude Segment

Offset	Field	Size	Value	Description
0	wLength	2	Number	Length of the Magnitude Response Segment, in bytes: $3+n*6$.
2	bSegmentType	1	Constant	CHANNEL_MAGNITUDE_RESPONSE.
3	dFreq(1)	4	Number	First frequency point.
7	wMagnitude(1)	2	Number	First magnitude value.
...	...	...	...	
$3+(n-1)*6$	dFreq(n)	4	Number	Last frequency point.
$7+(n-1)*6$	wMagnitude(n)	2	Number	Last magnitude value.

4.5.2.3.3.3.3 MAGNITUDE/PHASE RESPONSE

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

Table 4-24: Magnitude/Phase Segment

Offset	Field	Size	Value	Description
0	wLength	2	Number	Length of the Magnitude Response Segment, in bytes: $3+n*8$.
2	bSegmentType	1	Constant	CHANNEL_MAGNITUDE/PHASE_RESPONSE.
3	dFreq(1)	4	Number	First frequency point.
7	wMagnitude(1)	2	Number	First magnitude value.
9	wPhase(1)	2	Number	First phase value.
...	...	...	...	
$3+(n-1)*8$	dFreq(n)	4	Number	Last frequency point.
$7+(n-1)*8$	wMagnitude(n)	2	Number	Last magnitude value.
$9+(n-1)*8$	wPhase(n)	2	Number	Last phase value.

4.5.2.3.3.3.4 POSITION_XYZ SEGMENT

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

Table 4-25: Position Segment

Offset	Field	Size	Value	Description
0	wLength	2	Number	Length of the Segment, in bytes: 15.
2	bSegmentType	1	Constant	CHANNEL_POSITION_XYZ.
3	dX	4	Number	X-coordinate of the audio source or sink, associated with the channel.
7	dY	4	Number	Y-coordinate of the audio source or sink, associated with the channel.
11	dZ	4	Number	Z-coordinate of the audio source or sink, associated with the channel.

4.5.2.3.3.3.5 POSITION_RΘΦ SEGMENT

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

Table 4-26: Position Segment

Offset	Field	Size	Value	Description
0	wLength	2	Number	Length of the Segment, in bytes: 15.
2	bSegmentType	1	Constant	CHANNEL_POSITION_RΘΦ.
3	dR	4	Number	R-coordinate of the audio source or sink, associated with the channel.
7	dΘ	4	Number	Θ-coordinate of the audio source or sink, associated with the channel.
11	dΦ	4	Number	Φ-coordinate of the audio source or sink, associated with the channel.

4.5.2.3.3.3.6 VENDOR-DEFINED SEGMENT

Vendors are allowed to add vendor-defined Segments to the Channel Block to convey additional, proprietary channel information. The vendor-defined Segment shall use the following layout:

Table 4-27: Vendor-defined Segment

Offset	Field	Size	Value	Description
0	wLength	2	Number	Length of the End Segment, in bytes: 3+n.
2	bSegmentType	1	Constant	CHANNEL_VENDOR_DEFINED.
3	Vendor-defined	n	N/A	Vendor-defined extension of the Segment.

4.5.2.4 CONNECTORS DESCRIPTOR

The optional Connectors descriptor returns information about the physical attributes of all Connectors, associated with either an Input or an Output Terminal. The Connectors descriptor always uses the High Capability representation.

The **wDescriptorID** field contains an ID number that uniquely identifies the descriptor within the Audio Function. The value zero is reserved and shall not be used as a valid Connectors descriptor ID.

The **bNrConnectors** field indicates how many distinct physical Connectors are associated with this Terminal. Connectors are numbered from 1 to the total number of Connectors, associated with this Terminal. For each Connector, the descriptor contains a group of fields that further describe the Connector.

The **baConID(i)** field contains a unique identifier for Connector(i). The primary use for this is to indicate that the same Connector is associated with multiple Terminals. For example, one headset Connector incorporates the signals for the stereo headphone of the headset and also the signal for the mono microphone. This Connector would therefore be part of the Output Terminal that represents the stereo headphone and also be part of the Input Terminal that represents the microphone. This Connector would then be listed in both the Input Terminal and Output Terminal Connectors descriptor, using the same **baConID(i)** value in both descriptors to indicate the binding. Note that the **baConID(i)** value is never used as an identifier for an addressable Entity and therefore, **baConID(i)** values may overlap with Entity ID values.

The **waClusterDescrID(i)** contains the unique ID of a physical Cluster descriptor, describing the actual physical channels that are carried over Connector(i). Note that in most cases, the union of all the physical Cluster descriptors, referenced by the Connectors descriptor, will be identical to the logical Cluster descriptor, associated with the Pin of the Terminal. It is allowed for channels to appear in more than one physical Cluster descriptor to accommodate the situation where the same physical channels are available on different Connectors. For example, the same channel may be available on a balanced XLR connector as well as on an unbalanced BNC connector. For a detailed description of the Cluster descriptor, see Section 4.3, “Cluster Descriptor”.

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

The **bmaConAttributes(i)** field contains a bitmap that identifies the gender of Connector(i) (D1..0) and also indicates whether this Control is able to detect insertion and removal. If this bit is set for Connector(i), then bit D_{i-1} in the bitmap returned by the Insertion Control request indicates whether this Connector is inserted ($D_{i-1} = 1$) or not ($D_{i-1} = 0$). If this bit is clear for Connector(i), then bit D_{i-1} in the bitmap returned by the Insertion Control request has no meaning and shall be ignored.

The **waConDescrStr(i)** field is the ID of a String descriptor that contains a human-readable identifier for Connector(i). It is recommended that this string makes it easy to uniquely identify the Connector on the device’s chassis. This could be achieved by explicitly labeling the Connector on the device’s enclosure with the **waConDescrStr(i)** string content.

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

Table 4-28: Connectors Descriptor

Offset	Field	Size	Value	Description
0	wLength	2	Number	Size of this descriptor, in bytes: 7+n*11.
2	bDescriptorType	1	Constant	CS_INTERFACE descriptor type.
3	bDescriptorSubtype	1	Constant	CONNECTORS descriptor subtype.
4	wDescriptorID	2	Number	Unique ID of this Connectors descriptor.
6	bNrConnectors	1	Number	The number of Connectors associated with this Terminal: n
7	baConID(1)	1	Number	ID for Connector(1). Can be used to indicate that the same Connector is associated with multiple Terminals.
8	waClusterDescrID(1)	2	Number	ID of the physical Cluster descriptor transported over Connector(1).

Offset	Field	Size	Value	Description
10	baConType(1)	1	Constant	The Connector Type of Connector(1).
11	bmaConAttributes(1)	1	Bitmap	D1..0: Gender: 00: Gender Neutral. 01: Male. 10: Female. 11: Reserved. D2: Insertion/Removal Reporting: 0: No. 1: Yes. D7..3: Reserved.
12	waConDescrStr(1)	2	wStrDescrID	ID of a String descriptor that provides a physical description of Connector(1).
14	daConColor(1)	4	Number	The Connector color of Connector(1).
...	...	...	...	...
7+(n-1)*11	baConID(n)			Unique identifier for Connector(n). Can be used to indicate that the same Connector is associated with multiple Terminals.
8+(n-1)*11	waClusterDescrID(n)	2	Number	The unique ID of the physical Cluster descriptor transported over Connector(n).
10+(n-1)*11	baConType(n)	1	Constant	The Connector Type of Connector(n).
11+(n-1)*11	bmaConAttributes(n)	1	Bitmap	D1..0: Gender: 00: Gender Neutral. 01: Male. 10: Female. 11: Reserved. D2: Insertion/Removal Reporting: 0: No. 1: Yes. D7..3: Reserved.
12+(n-1)*11	waConDescrStr(n)	2	wStrDescrID	ID of a String descriptor that provides a physical description of Connector(n).
14+(n-1)*11	daConColor(n)	4	Number	The Connector color of Connector(n).

4.5.2.5 MIXER UNIT DESCRIPTOR

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

The **bNrInPins** field contains the number of Input Pins (*p*) of the Mixer Unit. This evidently equals the number of 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 Clusters.

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

The **wClusterDescrID** contains the unique ID of the Cluster descriptor that characterizes the Cluster that leaves the Mixer Unit over the single Output Pin ('downstream' connection). For a detailed description of the Cluster descriptor, see Section 4.3, "Cluster Descriptor".

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

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

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

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

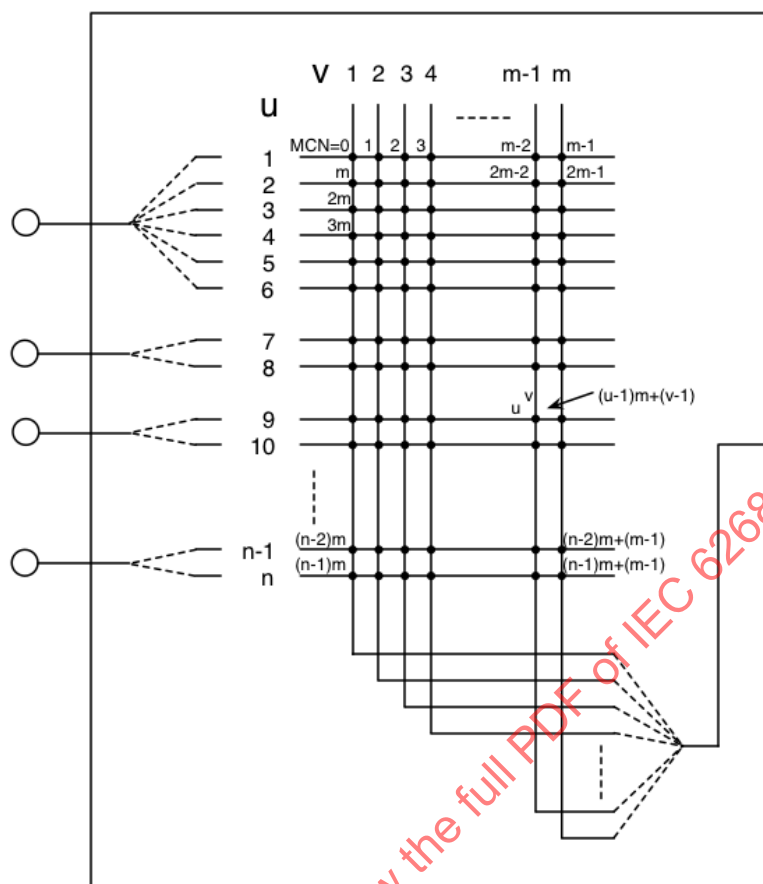
The Mixer Unit descriptor reports which Controls are programmable in the **bmMixerControls** bitmap field. This bitmap shall be interpreted as a two-dimensional bit array that has a row for each logical input channel and a column for each logical output channel. If a bit at position $[u, v]$ is set to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

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

$$MCN = (u - 1) \cdot m + (v - 1)$$

The following figure presents a more graphical explanation.

Figure 4-8: Mixer internals



The current setting of the Mixer Control (both programmable and fixed) at any position in the matrix can always be retrieved through the appropriate request. Therefore, the Mixer Unit shall always implement the Get request with the CUR attribute for each node in the matrix. See Section 5.2.1.7, “Mixer Unit Control Request” for further details.

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

```
IF ((n * m) MOD 8) <> 0 THEN
```

$$N = ((n * m) \text{ DIV } 8) + 1$$

ELSE

$$N = ((n * m) \text{ DIV } 8)$$

An ID of a String descriptor is provided to further describe the Mixer Unit.

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

Table 4-29: Mixer Unit Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	Number	Size of this descriptor, in bytes: 13+p+N.
1	bDescriptorType	1	Constant	CS_INTERFACE descriptor type.
2	bDescriptorSubtype	1	Constant	MIXER_UNIT descriptor subtype.
3	bUnitID	1	Number	Value 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	wClusterDescrID	2	Number	ID of the Cluster descriptor for this Mixer Unit.
7+p	bmMixerControls	N	Number	Bitmap indicating which Mixer Controls are programmable.
7+p+N	bmControls	4	Bitmap	D1..0: Underflow Control. D3..2: Overflow Control. D31..4: Reserved.
11+p+N	wMixerDescrStr	2	wStrDescrID	ID of a class-specific String descriptor, describing the Mixer Unit.

4.5.2.6 SELECTOR UNIT DESCRIPTOR

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

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

The Cluster descriptors, describing the logical channels that enter the Selector Unit are not repeated here. In order for a Selector Unit to be legally connected, *all* of the Clusters that enter the Selector Unit shall have exactly the same Cluster descriptor content.

An ID of a String descriptor is provided to further describe the Selector Unit.

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

Table 4-30: Selector Unit Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	Number	Size of this descriptor, in bytes: 11+p.
1	bDescriptorType	1	Constant	CS_INTERFACE descriptor type.
2	bDescriptorSubtype	1	Constant	SELECTOR_UNIT descriptor subtype.
3	bUnitID	1	Number	Value uniquely identifying the Unit within the Audio Function. This value is used in all requests to address this Unit.
4	bNrInPins	1	Number	Number of Input Pins of this Unit: p
5	baSourceID(1)	1	Number	ID of the Unit or Terminal to which the first Input Pin of this Selector Unit is connected.
...	...	...	...	...
5+(p-1)	baSourceID (p)	1	Number	ID of the Unit or Terminal to which the last Input Pin of this Selector Unit is connected.
5+p	bmControls	4	Bitmap	D1..0: Selector Control. D31..2: Reserved.
9+p	wSelectorDescrStr	2	wStrDescrID	ID of a class-specific String descriptor, describing the Selector Unit.

4.5.2.7 FEATURE UNIT DESCRIPTOR

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

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

An ID of 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 4-31: Feature Unit Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	Number	Size of this descriptor, in bytes: 7+(ch+1)*4
1	bDescriptorType	1	Constant	CS_INTERFACE descriptor type.
2	bDescriptorSubtype	1	Constant	FEATURE_UNIT descriptor subtype.
3	bUnitID	1	Number	Value uniquely identifying the Unit within the Audio Function. This value is used in all requests to address this Unit.
4	bSourceID	1	Number	ID of the Unit or Terminal to which this Feature Unit is connected.
5	bmaControls(0)	4	Bitmap	The Controls bitmap for master channel 0: D1..0: Mute Control. D3..2: Volume Control. D5..4: Bass Control. D7..6: Mid Control. D9..8: Treble Control. D11..10: Graphic Equalizer Control. D13..12: Automatic Gain Control. D15..14: Delay Control. D17..16: Bass Boost Control. D19..18: Loudness Control. D21..20: Input Gain Control. D23..22: Input Gain Pad Control. D25..24: Phase Inverter Control. D27..26: Underflow Control. D29..28: Overflow Control. D31..30: Reserved.
5+(1*4)	bmaControls(1)	4	Bitmap	The Controls bitmap for logical channel 1.
...	...	...	...	...
5+(ch*4)	bmaControls(ch)	4	Bitmap	The Controls bitmap for logical channel ch.
5+(ch+1)*4	wFeatureDescrStr	2	wStrDescrID	ID of a class-specific String descriptor, describing this Feature Unit.

4.5.2.8 SAMPLING RATE CONVERTER UNIT DESCRIPTOR

The Sampling Rate Converter Unit descriptor (RUD) provides information to the Host that is related to the functional aspects of the SRC Unit.

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

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

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

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

An ID of a String descriptor is provided to further describe the SRC Unit.

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

Table 4-32: Sampling Rate Converter Unit 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	SAMPLE_RATE_CONVERTER descriptor subtype.
3	bUnitID	1	Number	Value uniquely identifying the Unit within the Audio Function. This value is used in all requests to address this Unit.
4	bSourceID	1	Number	ID of the Unit or Terminal to which this SRC Unit is connected.
5	bCSourceInID	1	Number	ID of the Clock Entity to which this SRC Unit input section is connected.
6	bCSourceOutID	1	Number	ID of the Clock Entity to which this SRC Unit output section is connected.
7	wSRCDescrStr	2	wStrDescrID	ID of a class-specific String descriptor, describing the SRC Unit.

4.5.2.9 EFFECT UNIT DESCRIPTOR

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

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

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

bmaControls() is a (ch+1)-element array of 4-byte bitmaps, each following the same semantics as the **bmControls** field in other descriptors.

An ID of a String descriptor is provided to further describe the Effect Unit.

The following table outlines the Effect Unit descriptor.

Table 4-33: Effect Unit Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	Number	Size of this descriptor, in bytes: 17+(ch*4).
1	bDescriptorType	1	Constant	CS_INTERFACE descriptor type.
2	bDescriptorSubType	1	Constant	EFFECT_UNIT descriptor subtype.
3	bUnitID	1	Number	Value uniquely identifying the Unit within the Audio Function. This value is used in all requests to address this Unit.
4	wEffectType	2	Constant	Constant identifying the type of effect this Unit is performing.
6	bSourceID	1	Number	ID of the Unit or Terminal to which this Effect Unit is connected.
7	bmaControls(0)	4	Bitmap	The Controls bitmap for master channel 0: D31..0: Effect-specific allocation.
11	bmaControls(1)	4	Bitmap	The Controls bitmap for channel 1: D31..0: Effect-specific allocation.
...	...	...	...	...
11+(ch*4)	bmaControls(ch)	4	Bitmap	The Controls bitmap for channel ch: D31..0: Effect-specific allocation.
15+(ch*4)	wEffectsDescrStr	2	wStrDescrID	ID of a class-specific String descriptor, describing this Effect Unit.

4.5.2.9.1 PARAMETRIC EQUALIZER SECTION EFFECT UNIT DESCRIPTOR

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

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

The Enable Control shall always be present and implemented as Read-Write. It is therefore not represented in the **bmControls** field.

Table 4-34: Parametric Equalizer Section Effect Unit Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	Number	Size of this descriptor, in bytes: 17+(ch*4)
1	bDescriptorType	1	Constant	CS_INTERFACE descriptor type.
2	bDescriptorSubtype	1	Constant	EFFECT_UNIT descriptor subtype.
3	bUnitID	1	Number	Value uniquely identifying the Unit within the Audio Function. This value is used in all requests to address this Unit.
4	wEffectType	2	Constant	PARAM_EQ_SECTION_EFFECT effect type.
6	bSourceID	1	Number	ID of the Unit or Terminal to which this Effect Unit is connected.
7	bmaControls(0)	4	Bitmap	The Controls bitmap for master channel 0: D1..0: Center Frequency Control. D3..2: Q Factor Control. D5..4: Gain Control. D7..6: Underflow Control. D9..8: Overflow Control. D31..10: Reserved.
11	bmaControls(1)	4	Bitmap	The Controls bitmap for logical channel 1.
...	...	...	...	...
11+(ch*4)	bmaControls(ch)	4	Bitmap	The Controls bitmap for logical channel ch.
15+(ch*4)	wEffectsDescrStr	2	wStrDescrID	ID of a class-specific String descriptor, describing this Effect Unit.

4.5.2.9.2 REVERBERATION EFFECT UNIT DESCRIPTOR

The **wEffectType** field of the common Effect Unit descriptor contains the value REVERBERATION_EFFECT. (see Appendix A.19, “Effect Unit Effect Types.”)

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

The Enable Control shall always be present and implemented as Read-Write. It is therefore not represented in the **bmControls** field.

Table 4-35: Reverberation Effect Unit Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	Number	Size of this descriptor, in bytes: 17+(ch*4).
1	bDescriptorType	1	Constant	CS_INTERFACE descriptor type.
2	bDescriptorSubtype	1	Constant	EFFECT_UNIT descriptor subtype.
3	bUnitID	1	Number	Value uniquely identifying the Unit within the Audio Function. This value is used in all requests to address this Unit.
4	wEffectType	2	Constant	REVERBERATION_EFFECT effect type.
6	bSourceID	1	Number	ID of the Unit or Terminal to which this Effect Unit is connected.
7	bmaControls(0)	4	Bitmap	The Controls bitmap for master channel 0: D1..0: Type Control. D3..2: Level Control. D5..4: Time Control. D7..6: Delay Feedback Control. D9..8: Pre-Delay Control. D11..10: Density Control. D13..12: Hi-Freq Roll-Off Control. D15..14: Underflow Control. D17..16: Overflow Control. D31..18: Reserved.
11	bmaControls(1)	4	Bitmap	The Controls bitmap for logical channel 1.
...	...	...	...	...
11+(ch*4)	bmaControls(ch)	4	Bitmap	The Controls bitmap for logical channel ch.
15+(ch*4)	wEffectsDescrStr	2	wStrDescrID	ID of a class-specific String descriptor, describing this Effect Unit.

4.5.2.9.3 MODULATION DELAY EFFECT UNIT DESCRIPTOR

The **wEffectType** field of the common Effect Unit descriptor contains the value MOD_DELAY_EFFECT. (see Appendix A.19, “Effect Unit Effect Types.”)

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

Table 4-36: Modulation Delay Effect Unit Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	Number	Size of this descriptor, in bytes: 17+(ch*4).
1	bDescriptorType	1	Constant	CS_INTERFACE descriptor type.
2	bDescriptorSubtype	1	Constant	EFFECT_UNIT descriptor subtype.
3	bUnitID	1	Number	Value uniquely identifying the Unit within the Audio Function. This value is used in all requests to address this Unit.
4	wEffectType	2	Constant	MOD_DELAY_EFFECT effect type.
6	bSourceID	1	Number	ID of the Unit or Terminal to which this Effect Unit is connected.
7	bmaControls(0)	4	Bitmap	The Controls bitmap for master channel 0: D1..0: Enable Control. D3..2: Balance Control. D5..4: Rate Control. D7..6: Depth Control. D9..8: Time Control. D11..10: Feedback Level Control. D13..12: Underflow Control. D15..14: Overflow Control. D31..16: Reserved.
11	bmaControls(1)	4	Bitmap	The Controls bitmap for logical channel 1.
...	...	...	...	...
11+(ch*4)	bmaControls(ch)	4	Bitmap	The Controls bitmap for logical channel ch.
15+(ch*4)	wEffectsDescrStr	2	wStrDescrID	ID of a class-specific String descriptor, describing this Effect Unit.

4.5.2.9.4 DYNAMIC RANGE COMPRESSOR EFFECT UNIT DESCRIPTOR

The **wEffectType** field of the common Effect Unit descriptor contains the value DYN_RANGE_COMP_EFFECT. (see Appendix A.19, “Effect Unit Effect Types.”)

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

Table 4-37: Dynamic Range Compressor Effect Unit Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	Number	Size of this descriptor, in bytes: 17+(ch*4).
1	bDescriptorType	1	Constant	CS_INTERFACE descriptor type.
2	bDescriptorSubtype	1	Constant	EFFECT_UNIT descriptor subtype.
3	bUnitID	1	Number	Value uniquely identifying the Unit within the Audio Function. This value is used in all requests to address this Unit.
4	wEffectType	2	Constant	DYN_RANGE_COMP_EFFECT effect type.
6	bSourceID	1	Number	ID of the Unit or Terminal to which this Effect Unit is connected.
7	bmaControls(0)	4	Bitmap	The Controls bitmap for master channel 0: D1..0: Enable Control. D3..2: Compression Ratio Control. D5..4: MaxAmpl Control. D7..6: Threshold Control. D9..8: Attack Time Control. D11..10: Release Time Control. D13..12: Underflow Control. D15..14: Overflow Control. D31..16: Reserved.
11	bmaControls(1)	4	Bitmap	The Controls bitmap for logical channel 1.
...	...	...	...	...
11+(ch*4)	bmaControls(ch)	4	Bitmap	The Controls bitmap for logical channel ch.
15+(ch*4)	wEffectsDescrStr	2	wStrDescrID	ID of a class-specific String descriptor, describing this Effect Unit.

4.5.2.10 PROCESSING UNIT DESCRIPTOR

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

The **wProcessType** field contains a value that fully identifies the Processing Unit. For a list of all supported Processing Unit Types, see Appendix A.20, “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 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 **wClusterDescrID** field contains the unique ID of the Cluster descriptor that characterizes the Cluster that leaves the Processing Unit over the single Output Pin (‘downstream’ connection). For a detailed description of the Cluster descriptor, see Section 4.3, “Cluster Descriptor”.

A Processing Unit shall not support an Enable Control. If the functionality of the Audio Function requires that the Processing Unit can be bypassed in certain scenarios, then the topology of the Function shall make this explicit through the use of a Selector Unit where one Input Pin of the Selector Unit is connected to the Output Pin of the Processing Unit and the other Input Pin of the Selector Unit is connected to an Entity that provides an output channel Cluster that is compatible with the output Cluster of the Processing Unit. This may be the same Output Pin to which the Input Pin of the Processing Unit is connected (direct bypass) if the Cluster configuration is not altered by the Processing Unit.

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

An ID of 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 4-38: Common Part of the Processing Unit Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	Number	Size of this descriptor, in bytes: 9+p+x.
1	bDescriptorType	1	Constant	CS_INTERFACE descriptor type.
2	bDescriptorSubtype	1	Constant	PROCESSING_UNIT descriptor subtype.
3	bUnitID	1	Number	Value uniquely identifying the Unit within the Audio Function. This value is used in all requests to address this Unit.
4	wProcessType	2	Constant	Constant identifying the type of processing this Unit is performing.
6	bNrInPins	1	Number	Number of Input Pins of this Unit: p
7	baSourceID(1)	1	Number	ID of the Unit or Terminal to which the first Input Pin of this Processing Unit is connected.
...	...	...	...	...
7+(p-1)	baSourceID (p)	1	Number	ID of the Unit or Terminal to which the last Input Pin of this Processing Unit is connected.
7+p	wProcessingDescrStr	2	wStrDescrID	ID of a class-specific String descriptor, describing this Processing Unit.
9+p	Process-specific	x	N/A	A process-specific descriptor is appended to the common descriptor. See the following paragraphs.

4.5.2.10.1 UP/DOWN-MIX PROCESSING UNIT DESCRIPTOR

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

The Up/Down-mix Processing Unit has a single Input Pin. Therefore, the **bNrInputs** field shall contain the value 1.

The Mode Select Control (D1..0) 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 Select Control Request. The number of supported modes (m) is contained in the **bNrModes** field. This field is followed by an array of Cluster descriptor ID fields, **waClusterDescrID()**. The index i into this array is one-based and directly related to the number of the mode described by entry **waClusterDescrID(i)**. It is the value i that shall be used as a parameter for the Set Mode request to select the mode i .

Each **waClusterDescrID(i)** field contains the unique ID of a Cluster descriptor that describes the output Cluster configuration when mode i is selected. The Host changing the mode via a Set Mode Select Control request results in a change of output Cluster descriptor. However, this shall not lead to a High Capability Descriptor interrupt as the Host itself has initiated the change.

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

Table 4-39: Up/Down-mix Processing Unit Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	Number	Size of this descriptor, in bytes: $15+2*m$.
1	bDescriptorType	1	Constant	CS_INTERFACE descriptor type.
2	bDescriptorSubtype	1	Constant	PROCESSING_UNIT descriptor subtype.
3	bUnitID	1	Number	Value 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	baSourceID(1)	1	Number	ID of the Unit or Terminal to which the Input Pin of this Processing Unit is connected.
8	wProcessingDescrStr	2	wStrDescrID	ID of a class-specific String descriptor, describing this Processing Unit.
10	bmControls	4	Bitmap	D1..0: Mode Select Control. D3..2: Underflow Control. D5..4: Overflow Control. D31..6: Reserved.
14	bNrModes	1	Number	Number of modes, supported by this Processing Unit: m
15	waClusterDescrID(1)	2	Number	Unique ID of the Cluster descriptor for mode(1).
...	...	...	...	...
$15+2*(m-1)$	waClusterDescrID(m)	2	Number	Unique ID of the Cluster descriptor for mode(m).

4.5.2.10.2 STEREO EXTENDER PROCESSING UNIT DESCRIPTOR

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

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

The input Cluster to the Stereo Extender Processing Unit shall contain only the Left and Right logical input channels. The output Cluster shall therefore also only contain the Left and Right logical channels. As a consequence, there is no need for an output Cluster descriptor for this Processing Unit.

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

The following table outlines the Stereo Extender Processing Unit descriptor.

Table 4-40: Stereo Extender Processing Unit Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	Number	Size of this descriptor, in bytes: 14.
1	bDescriptorType	1	Constant	CS_INTERFACE descriptor type.
2	bDescriptorSubtype	1	Constant	PROCESSING_UNIT descriptor subtype.
3	bUnitID	1	Number	Value uniquely identifying the Unit within the Audio Function. This value is used in all requests to address this Unit.
4	wProcessType	2	Constant	STEREO_EXTENDER_PROCESS process type.
6	bNrInPins	1	Number	Number of Input Pins of this Unit: 1
7	baSourceID(1)	1	Number	ID of the Unit or Terminal to which the Input Pin of this Processing Unit is connected.
8	wProcessingDescrStr	2	wStrDescrID	ID of a class-specific String descriptor, describing this Processing Unit.
10	bmControls	4	Bitmap	D1..0: Width Control. D3..2: Underflow Control. D5..4: Overflow Control. D31..6: Reserved.

4.5.2.10.3 MULTI-FUNCTION PROCESSING UNIT DESCRIPTOR

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

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

Because a Multi-Function Processing Unit can freely redefine its output Cluster configuration, there is a need for an output Cluster descriptor. The **wClusterDescrID** field contains the unique ID of the Cluster descriptor that characterizes the Cluster that leaves the Processing Unit over its single Output Pin (‘downstream’ connection). For a detailed description of the Cluster descriptor, see Section 4.3, “Cluster Descriptor”.

The **bmAlgorithms** field is a bitmap that indicates what types of algorithms are performed inside the Multi-Function Processing Unit. Multiple bits may be set simultaneously.

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

Table 4-41: Multi-Function Processing Unit Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	Number	Size of this descriptor, in bytes: 19+p.
1	bDescriptorType	1	Constant	CS_INTERFACE descriptor type.
2	bDescriptorSubtype	1	Constant	PROCESSING_UNIT descriptor subtype.
3	bUnitID	1	Number	Value uniquely identifying the Unit within the Audio Function. This value is used in all requests to address this Unit.
4	wProcessType	2	Constant	MULTI_FUNCTION_PROCESS process type.
6	bNrInPins	1	Number	Number of Input Pins of this Unit: 1
7	baSourceID(1)	1	Number	ID of the Unit or Terminal to which Input Pin 1 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	wProcessingDescrStr	2	wStrDescrID	ID of a class-specific String descriptor, describing this Processing Unit.
9+p	bmControls	4	Bitmap	D1..0: Underflow Control. D3..2: Overflow Control. D31..4: Reserved.
13+p	wClusterDescrID	2	Number	Unique ID of the Cluster descriptor for this Processing Unit.
15+p	bmAlgorithms	4	Bitmap	D0: Algorithm Undefined. D1: Beam Forming. D2: Acoustic Echo Cancellation. D3: Active Noise Cancellation. D4: Blind Source Separation. D5: Noise Suppression/Reduction. D31..6: Reserved.

4.5.2.11 EXTENSION UNIT DESCRIPTOR

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

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

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

Because an Extension Unit can freely redefine its output Cluster configuration, there is a need for an output Cluster descriptor. The **wClusterDescrID** field contains the unique ID of the Cluster descriptor that characterizes the Cluster that leaves the Extension Unit over its single Output Pin ('downstream' connection). For a detailed description of the Cluster descriptor, see Section 4.3, "Cluster Descriptor".

An ID of a String descriptor is provided to further describe the Extension Unit.

The following table outlines the Extension Unit descriptor.

Table 4-42: Extension Unit Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	Number	Size of this descriptor, in bytes: 15+p.
1	bDescriptorType	1	Constant	CS_INTERFACE descriptor type.
2	bDescriptorSubtype	1	Constant	EXTENSION_UNIT descriptor subtype.
3	bUnitID	1	Number	Value 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	wExtensionDescrStr	2	wStrDescrID	ID of a class-specific String descriptor, describing this Extension Unit.
9+p	bmControls	4	Bitmap	D1..0: Underflow Control. D3..2: Overflow Control. D31..4: Reserved.
13+p	wClusterDescrID	2	Number	Unique ID of the Cluster descriptor for this Extension Unit.

4.5.2.12 CLOCK SOURCE DESCRIPTOR

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

The **bmAttributes** field contains a Clock Type bit field (D0) that indicates whether the Clock Source represents an external clock (D0 = 0b0) or an internal clock (D0=0b1). Further characteristics of the Clock Source can be derived from the value of the Clock Frequency Control bit pair D1..0 in the **bmControls** field. Since the Clock Frequency Control shall always be present, the only allowed values are D1..0 = 0b01 (Read-Only) or D1..0 = 0b11 (Read-Write). The supported clock frequencies can be derived from the Range attribute of the Clock Frequency Control (fixed rate vs. variable rate). Note that even a Clock Source of Type External can be Read-Write if the Audio Function has the ability to influence that external clock through means outside of USB. The actual sampling frequency of the Clock Source can be manipulated through the Clock Frequency request. In addition, the Clock Source can be queried for the validity of its current sampling clock signal through a Get Clock Validity request.

Bit D1 in the **bmAttributes** field indicates whether an internal clock is free running (D1 = 0b0) or synchronized to the Start of Frame (D1 = 0b1). If D0 = 0b0, then D1 shall also be set to 0b0.

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

An ID of a String descriptor is provided to further describe the Clock Source Unit.

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

Table 4-43: Clock Source 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	CLOCK_SOURCE descriptor subtype.
3	bClockID	1	Number	Value uniquely identifying the Clock Source Entity within the Audio Function. This value is used in all requests to address this Entity.
4	bmAttributes	1	Bitmap	D0: Clock Type: 0: External Clock. 1: Internal Clock. D1: Synchronization Type: 0: Asynchronous. 1: Clock synchronized to SOF. D7..2: Reserved.
5	bmControls	4	Bitmap	D1..0: Clock Frequency Control. D3..2: Clock Validity Control. D31..4: Reserved.
9	bReferenceTerminal	1	Number	Terminal ID of the Terminal from which this Clock Source is derived.
10	wClockSourceStr	2	wStrDescrID	ID of a class-specific String descriptor, describing the Clock Source Entity.

4.5.2.13 CLOCK SELECTOR DESCRIPTOR

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

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

An ID of a String descriptor is provided to further describe the Clock Selector Entity.

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

Table 4-44: Clock Selector Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	Number	Size of this descriptor, in bytes: 11+p.
1	bDescriptorType	1	Constant	CS_INTERFACE descriptor type.
2	bDescriptorSubtype	1	Constant	CLOCK_SELECTOR descriptor subtype.
3	bClockID	1	Number	Value uniquely identifying the Clock Selector Entity within the Audio Function. This value is used in all requests to address this Entity.
4	bNrInPins	1	Number	Number of Input Pins of this Unit: p
5	baCSourceID(1)	1	Number	ID of the Clock Entity to which the first Clock Input Pin of this Clock Selector Entity is connected.
...	...	...	...	...
5+(p-1)	baCSourceID (p)	1	Number	ID of the Clock Entity to which the last Clock Input Pin of this Clock Selector Entity is connected.
5+p	bmControls	4	Bitmap	D1..0: Clock Selector Control. D31..2: Reserved.
9+p	wCSelectorDescrStr	2	wStrDescrID	ID of a class-specific String descriptor, describing the Clock Selector Entity.

4.5.2.14 CLOCK MULTIPLIER DESCRIPTOR

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

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

An ID of a String descriptor is provided to further describe the Clock Multiplier Entity.

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

Table 4-45: Clock Multiplier Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	Number	Size of this descriptor, in bytes: 11.
1	bDescriptorType	1	Constant	CS_INTERFACE descriptor type.
2	bDescriptorSubtype	1	Constant	CLOCK_MULTIPLIER descriptor subtype.
3	bClockID	1	Number	Value uniquely identifying the Clock Multiplier Entity within the Audio Function. This value is used in all requests to address this Entity.
4	bCSourceID	1	Number	ID of the Clock Entity to which the last Clock Input Pin of this Clock Selector Entity is connected.
5	bmControls	4	Bitmap	D1..0: Clock Numerator Control. D3..2: Clock Denominator Control. D31..4: Reserved.

Offset	Field	Size	Value	Description
9	wCMultiplierDescrStr	2	wStrDescrID	ID of a class-specific String descriptor, describing the Clock Multiplier Entity.

4.5.2.15 POWER DOMAIN DESCRIPTOR

The Power Domain descriptor (PDD) provides information to the Host regarding the existence of one or more Power Domains within the Audio Function and lists the Entities, through their respective Entity ID, that are explicit members of a particular Power Domain. Only Input Terminals, Output Terminals, Effect Units, Processing Units, and Extension Units shall be explicit members of a Power Domain.

There is a Power Domain descriptor for each Power Domain in the Audio Function. Therefore, the number of Power Domain descriptors is an indicator for the number of separately managed Power Domains in the Audio Function. Each eligible Entity can only be member of a single Power Domain, i.e. Power Domains never overlap.

The Power Domain is uniquely identified by the value in the **bPowerDomainID** field of the Power Domain descriptor (PDD). No other Entity within the AudioControl interface may have the same ID. This value shall be passed in the Entity ID field (part of the **wIndex** field) of each request that is directed to the Power Domain.

The **waRecoveryTime()** array contains the approximate recovery time for Power Domain State D1 and D2. The **bNrEntities** field contains the number of Entities in this Power Domain. The **baEntityID()** array contains the Entity IDs of all the explicit member Entities.

An ID of a String descriptor is provided to further describe the Power Domain.

Table 4-46: Power Domain Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	Number	Size of this descriptor, in bytes: 11+p.
1	bDescriptorType	1	Constant	CS_INTERFACE descriptor type.
2	bDescriptorSubtype	1	Constant	POWER_DOMAIN descriptor subtype.
3	bPowerDomainID	1	Number	Value uniquely identifying the Power Domain within the Audio Function. This value is used in all requests to address this Power Domain.
4	waRecoveryTime(1)	2	Number	Time to recover from D1 to D0. Expressed in 50 μ s increments. For example, a typical value for this is 600, which indicates a recovery time of 30 ms.
6	waRecoveryTime(2)	2	Number	Time to recover from D2 to D0. Expressed in 50 μ s increments. For example, a typical value for this is 6000, which indicates a recovery time of 300 ms.
8	bNrEntities	1	Number	Number of Entities belonging to this Power Domain: p.
9	baEntityID(1)	1	Number	ID of the first Entity that belongs to this Power Domain.
...	...	...	...	
8+p	baEntityID(p)	1	Number	ID of the last Entity that belongs to this Power Domain.
9+p	wPDomainDescrStr	2	wStrDescrID	ID of a class-specific String descriptor, describing this Power Domain.

4.6 AUDIOCONTROL ENDPOINT DESCRIPTORS

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

4.6.1 AC CONTROL ENDPOINT DESCRIPTORS

4.6.1.1 STANDARD AC CONTROL ENDPOINT DESCRIPTOR

The AudioControl interface uses the default endpoint 0. Therefore, there is no dedicated standard control endpoint descriptor.

4.6.1.2 CLASS-SPECIFIC AC CONTROL ENDPOINT DESCRIPTOR

There is no dedicated class-specific control endpoint descriptor.

4.6.2 AC INTERRUPT ENDPOINT DESCRIPTORS

4.6.2.1 STANDARD AC INTERRUPT ENDPOINT DESCRIPTOR

The interrupt endpoint descriptor is identical to the standard endpoint descriptor defined in the *USB 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 4-47: Standard AC Interrupt Endpoint Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	Number	Size of this descriptor, in bytes: 7
1	bDescriptorType	1	Constant	ENDPOINT descriptor type
2	bEndpointAddress	1	Endpoint	The address of the endpoint on the USB device described by this descriptor. The address is encoded as follows: D7: Direction. 1 = IN endpoint. D6..4: Reserved. D3..0: The endpoint number, determined by the designer.
3	bmAttributes	1	Bitmap	D1..0: Transfer Type 11 = Interrupt. All other bits are reserved.
4	wMaxPacketSize	2	Number	Maximum packet size this endpoint is capable of sending or receiving when this configuration is selected. Used here to pass 6-byte interrupt information.
6	bInterval	1	Number	Interval for polling the Interrupt endpoint.

4.6.2.2 CLASS-SPECIFIC AC INTERRUPT ENDPOINT DESCRIPTOR

There is no class-specific AudioControl interrupt endpoint descriptor.

4.7 AUDIOSTREAMING INTERFACE DESCRIPTORS

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

4.7.1 STANDARD AS INTERFACE DESCRIPTOR

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

Table 4-48: 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 the interface. A zero-based value identifying the index in the array of concurrent interfaces supported by this configuration.
3	bAlternateSetting	1	Number	Value used to select an Alternate Setting for the interface identified in the prior field.
4	bNumEndpoints	1	Number	Number of endpoints used by this interface (excluding endpoint 0). Shall be either 0 (no data endpoint), 1 (data endpoint) or 2 (data and explicit feedback endpoint).
5	bInterfaceClass	1	Class	AUDIO Audio Interface Class code (assigned by the USB). See Appendix A.4, “Audio Interface Class Code”
6	bInterfaceSubClass	1	Subclass	AUDIO_STREAMING Audio Interface Subclass code. Assigned by this specification. See Appendix A.5, “Audio Interface Subclass Codes.”
7	bInterfaceProtocol	1	Protocol	IP_VERSION_03_00 Interface Protocol code. Indicates the current version of the specification. See Appendix A.6, “Audio Interface Protocol Codes”
8	iInterface	1	Index	Index of a String descriptor that describes this interface.

4.7.2 CLASS-SPECIFIC AS INTERFACE DESCRIPTOR

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

The **wClusterDescrID** field contains the unique ID of the cluster descriptor that characterizes the physical cluster associated with an Alternate Setting of this interface. For a detailed description of the cluster descriptor, see Section 4.3, “Cluster Descriptor”.

The remaining fields in this descriptor are described in detail in a separate document, *USB Audio Data Formats* that is considered part of this specification.

An Alternate Setting of an interface is allowed to support multiple Audio Data Formats at the same time, even if they belong to different Format Types. The **bmFormats** bitmap has a bit set for each Audio Data Format that can be used when communicating with this Alternate Setting of the interface. It is up to the implementation to be able to accurately distinguish among the different Audio Data Formats and invoke the correct encoding or decoding processes for the current Audio Data Format. Alternatively, support for the different Format Types and/or Audio Data Formats can be separated out into different Alternate Settings if the interface is not able to accurately distinguish among the different Format Types and/or Audio Data Formats.

Table 4-49: Class-Specific AS Interface Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	Number	Size of this descriptor in bytes: 23.
1	bDescriptorType	1	Constant	CS_INTERFACE descriptor type.
2	bDescriptorSubtype	1	Constant	AS_GENERAL descriptor subtype.
3	bTerminalLink	1	Number	The Terminal ID of the Terminal to which this interface is connected.
4	bmControls	4	Bitmap	D1..0: Active Alternate Setting Control. D3..2: Valid Alternate Settings Control. D5..4: Audio Data Format Control. D31..6: Reserved.
8	wClusterDescrID	2	Number	ID of the cluster descriptor of the AS Interface.
10	bmFormats	8	Bitmap	The Audio Data Format(s) that can be used to communicate with this interface. See the <i>USB Audio Data Formats</i> document for further details.
18	bSubslotSize	1	Number	The number of bytes occupied by one audio subslot.
19	bBitResolution	1	Number	The number of effectively used bits from the available bits in an audio subslot.
20	bmAuxProtocols	2	Bitmap	Bitmap, indicating which Auxiliary Protocols are required.
22	bControlSize	1	Number	Size of the Control Channel Words, in bytes.

4.7.3 CLASS-SPECIFIC AS VALID FREQUENCY RANGE DESCRIPTOR

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

Table 4-50: Class-Specific AS Valid Frequency Range Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	Number	Size of this descriptor in bytes: 11
1	bDescriptorType	1	Constant	CS_INTERFACE descriptor type.
2	bDescriptorSubtype	1	Constant	AS_VALID_FREQ_RANGE descriptor subtype.
3	dMin	4	Number	The minimum sampling frequency at which this Alternate Setting of the AudioStreaming interface is valid.
7	dMax	4	Number	The maximum sampling frequency at which this Alternate Setting of the AudioStreaming interface is valid.

4.8 AUDIOSTREAMING ENDPOINT DESCRIPTORS

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

4.8.1 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.

4.8.1.1 STANDARD AS ISOCHRONOUS AUDIO DATA ENDPOINT DESCRIPTOR

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

Table 4-51: Standard AS Isochronous Audio Data Endpoint Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	Number	Size of this descriptor, in bytes: 7
1	bDescriptorType	1	Constant	ENDPOINT descriptor type
2	bEndpointAddress	1	Endpoint	The address of the endpoint on the USB device described by this descriptor. The address is encoded as follows: D3..0: The endpoint number, determined by the designer. D6..4: Reserved. D7: Direction: 0 = OUT endpoint. 1 = IN endpoint.
3	bmAttributes	1	Bitmap	D1..0: Transfer type: 01 = Isochronous. D3..2: Synchronization Type: 01 = Asynchronous. 10 = Adaptive. 11 = Synchronous. D5..4: Usage Type: 00 = Data endpoint or 10 = Implicit feedback Data endpoint. All other bits are reserved.
4	wMaxPacketSize	2	Number	(Speed-dependent.) Indicates the maximum packet size and potentially any bursting on this endpoint. This is determined by the audio bandwidth constraints of the endpoint.
6	bInterval	1	Number	Interval for polling endpoint for data transfers.

Note: For SuperSpeed and SuperSpeedPlus endpoints, the SuperSpeed Endpoint Companion and SuperSpeedPlus Endpoint Companion descriptors would follow the standard endpoint descriptor. See the *USB 3.1 specification* for details.

4.8.1.2 CLASS-SPECIFIC AS ISOCRONOUS AUDIO DATA ENDPOINT DESCRIPTOR

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

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

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

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

Table 4-52: Class-Specific AS Isochronous Audio Data Endpoint Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	Number	Size of this descriptor, in bytes: 10.
1	bDescriptorType	1	Constant	CS_ENDPOINT descriptor type.
2	bDescriptorSubtype	1	Constant	EP_GENERAL descriptor subtype.
3	bmControls	4	Bitmap	D1..0: Pitch Control. D3..2: Data Overrun Control. D5..4: Data Underrun Control. D31..6: Reserved.
7	bLockDelayUnits	1	Number	Indicates the units used for the wLockDelay field: 0: Undefined. 1: Milliseconds. 2: Decoded PCM samples. 3..255: Reserved.
8	wLockDelay	2	Number	Indicates the time it takes this endpoint to reliably lock its internal clock recovery circuitry. Units used depend on the value of the bLockDelayUnits field.

4.8.2 AS ISOCRONOUS FEEDBACK ENDPOINT DESCRIPTOR

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

4.8.2.1 STANDARD AS ISOCRONOUS FEEDBACK ENDPOINT DESCRIPTOR

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

Table 4-53: Standard AS Isochronous Feedback Endpoint Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	Number	Size of this descriptor, in bytes: 7
1	bDescriptorType	1	Constant	ENDPOINT descriptor type.
2	bEndpointAddress	1	Endpoint	The address of the endpoint on the USB device described by this descriptor. The address is encoded as follows: D3..0: The endpoint number, determined by the designer. D6..4: Reserved. D7: Direction: 0 = OUT endpoint. 1 = IN endpoint.
3	bmAttributes	1	Bitmap	D1..0: Transfer type: 01 = Isochronous. D3..2: Synchronization Type: 00 = No Synchronization. D5..4: Usage Type: 01 = Feedback endpoint. All other bits are reserved.
4	wMaxPacketSize	2	Number	Maximum packet size this endpoint is capable of sending or receiving when this configuration is selected.
6	bInterval	1	Number	Interval for polling endpoint for data transfers.

Note: For SuperSpeed and SuperSpeedPlus endpoints, the SuperSpeed Endpoint Companion and SuperSpeedPlus Endpoint Companion descriptors would follow the standard endpoint descriptor. See the *USB 3.1 specification* for details.

4.8.2.2 CLASS-SPECIFIC AS ISOCRONOUS FEEDBACK ENDPOINT DESCRIPTOR

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

4.9 CLASS-SPECIFIC STRING DESCRIPTORS

This specification defines a new type of class-specific String descriptor. Class-specific String descriptors are retrieved through a class-specific Get String request. See Section 5.2.3.2, “Class-specific String Request” for details. All class-specific strings in the Audio Function shall use this new methodology. Strings that are part of the standard descriptor set shall use the standard string methodology.

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

The **wLength** field contains the length of the class-specific String descriptor. Class-specific strings can be up to 65,528 bytes in length.

The **bDescriptorType** field shall be set to CS_STRING.

The **bDescriptorSubtype** field indicates the descriptor subtype for the String descriptor. (Currently, only the value SUBTYPE_UNDEFINED is defined.)

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

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

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

Table 4-54: Class-specific String Descriptor

Offset	Field	Size	Value	Description
0	wLength	2	Number	Size of the String descriptor in bytes: 7+N.
2	bDescriptorType	1	Number	CS_STRING. Type of this descriptor.
3	bDescriptorSubtype	1	Constant	Descriptor Subtype.
4	wStrDescrID	2	Number	Unique ID for this class-specific String descriptor.
6	iLangID	1	Number	Zero-based index into the LANGID code array as returned by the device.
7	String	N	Number	Unicode encoded string. Follows the definitions outlined in the <i>USB Specification</i> .

5 REQUESTS

5.1 STANDARD REQUESTS

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

5.2 CLASS-SPECIFIC REQUESTS

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

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

The Audio Device Class supports three additional class-specific requests:

- **Memory Requests.** Every addressable Entity in the Audio Function (Clock Entity, Terminal, Unit, Power Domain, Interface, and Endpoint) can expose a memory-mapped interface that provides the means to generically manipulate the Entity. Vendor-specific Control implementations could be based on this type of request.
- **String Requests.** This type of request provides a class-specific method to retrieve String descriptors from the Audio Function. This request is introduced to overcome the USB core specification limitation that only provides for 255 device-wide String descriptors.
- **Descriptor Requests.** This type of request provides a class-specific method to retrieve descriptors from the Audio Function outside the standard descriptor retrieval during enumeration. This request is introduced to overcome the USB core specification limitation that only provides for descriptors that are a maximum of 256 bytes long. It also enables dynamically changing descriptors (after enumeration).

In principle, all Controls and their associated requests are optional. If an Audio Function does not support a certain request, it shall 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 shall also be supported. Get requests may be supported without the associated Set request being supported. If interrupts are supported, then all necessary Get requests shall be implemented that are required to retrieve the appropriate information from the Audio Function in response to these interrupts.

The remainder of this section describes the class-specific requests and their characteristics used to manipulate the incorporated Controls, memory locations, class-specific strings and descriptors. Unless explicitly stated otherwise, all Controls are optional and shall be Read/Write, if present.

5.2.1 AUDIOCONTROL REQUESTS

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

5.2.1.1 CONTROL ATTRIBUTES

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

- Current setting attribute (CUR)
- Range attribute (RANGE)
- Interrupt Enable attribute (INTEN)

The CUR attribute is used to manipulate the current actual setting of a Control. The RANGE attribute provides information about the limitations the Control imposes on the allowed settings of the CUR attribute. The RANGE attribute actually consists of an array of sub-attributes. The sub-attributes are Minimum (MIN), Maximum (MAX), and Resolution (RES). They are always manipulated in triplets of the form [MIN, MAX, RES] and cannot be accessed or modified individually. The RANGE attribute supports an array of these triplets so that discontinuous multiple subranges of a Control can be accurately reported. The first element in the array contains the number of subranges the Control supports. Subsequent triplet elements in the array correspond to each of the subranges. The subranges shall be ordered in ascending order (from lower values to higher values). Individual subranges cannot overlap (i.e. the MAX value of the previous subrange cannot be equal to the MIN value of the next subrange). If a subrange consists of only a single value, the corresponding triplet shall contain that value for both its MIN and MAX sub-attribute and the RES sub-attribute shall be set to zero.

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

- $-\infty$ dB
- -70 dB to -40 dB in steps of 3 dB
- -40 dB to -20 dB in steps of 2 dB
- -20 dB to 0 dB in steps of 1 dB

One possible layout of the RANGE attribute is then:

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

Another way of representing the same Control is as follows:

```
RANGE(0) = 3
RANGE(1) = [-70, -43, 3]
RANGE(2) = [-40, -22, 2]
RANGE(3) = [-20, 0, 1]
```

It is left to the designer to choose a suitable representation.

The Interrupt Enable attribute is used to manipulate the Control's ability to generate an interrupt when any of its other attributes change other than through Host manipulation. The Interrupt Enable attribute shall be supported for all Controls that are able to generate an interrupt. When supported, the Interrupt Enable attribute is always Read-Write and therefore, both the Get and Set request shall be supported. The Interrupt

Enable attribute can take only two values: 0 (False, Disabled) or 1 (True, Enabled). All Interrupt Enable attributes shall have a default value of 1 (Enabled).

5.2.1.2 CONTROL REQUEST LAYOUT

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

Table 5-1: Request Layout

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

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

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

- Current setting attribute (CUR)
- Range attribute (RANGE)
- Interrupt Enable attribute (INTEN)

If the addressed Control does not support modification of a certain attribute, the control pipe shall indicate a stall when an attempt is made to modify that attribute. In most cases, only the CUR attribute will be supported for the Set request. However, this specification does not prevent a designer from making the RANGE attribute programmable or having the Audio Function adjust a RANGE attribute due to an external event. For the list of Request constants, refer to Appendix A.22, “Audio Class-Specific Request Codes.”

Note: Support for the INTEN attribute is not repeated for each Control in the following sections. It is assumed that support is provided as appropriate for the Control in a particular Audio Function implementation. Issuing a Get INTEN Request will result in a stall on the control pipe when the INTEN attribute is not supported for that Control.

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

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

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

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

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

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

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

The actual parameter(s) for the Get request are returned in the data stage of the control transfer. The length of the parameter block to return is indicated in the **wLength** field of the request. If the parameter block is longer than what is indicated in the **wLength** field, only the initial bytes of the parameter block are returned. If the parameter block is shorter than what is indicated in the **wLength** field, the device indicates the end of the control transfer by sending a short packet when further data is requested. The layout of the parameter block is qualified by both the **bRequest** and **wIndex** fields. Refer to the following sections for a detailed description of the parameter block layout for all possible Entities.

5.2.1.3 CONTROL REQUEST PARAMETER BLOCK LAYOUT

With a few exceptions, almost all Control requests manipulate a single Control parameter during a Set or Get request. For those requests, the possible parameter block layouts can be divided into three categories, depending on the byte size of the Control's CUR attribute. A CUR attribute's size can be either a single byte, a word (2 bytes) or a double word (4 bytes). The following paragraphs specify the layout of the CUR and RANGE parameter blocks for the three categories. The layout of the parameter block for the INTEN attribute is the same for all Controls and is specified in Section 5.2.1.3.4, "INTEN Parameter Block".

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

5.2.1.3.1 LAYOUT 1 PARAMETER BLOCK

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

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

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

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

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

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

5.2.1.3.2 LAYOUT 2 PARAMETER BLOCK

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

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

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

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

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

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

wLength		2+6*n		
Offset	Field	Size	Value	Description
6+6*(n-1)	wRES(n)	2	Number	The setting for the RES attribute of the last subrange of the addressed Control

5.2.1.3.3 LAYOUT 3 PARAMETER BLOCK

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

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

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

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

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

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

5.2.1.3.4 INTEN PARAMETER BLOCK

The parameter block for the INTEN attribute of a Control is as follows:

Table 5-8: INTEN Parameter Block

wLength		1		
Offset	Field	Size	Value	Description
0	bINTEN	1	Boolean	The setting for the INTEN attribute of the addressed Control. Only allowed values are 0 (False) and 1 (True).

5.2.1.4 COMMON CONTROLS

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

5.2.1.4.1 ENABLE CONTROL

The Enable Control is used to either enable the functionality of an Entity or bypass the Entity entirely. In the latter case, the input Cluster is routed unaltered to the output of the Entity. The Enable Control shall have only the CUR attribute. The value of an Enable Control CUR attribute can be either TRUE or FALSE.

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

The parameter block for this Control request uses Layout 1 (See Section 5.2.1.3.1, “Layout 1 Parameter Block.”)

5.2.1.4.2 UNDERFLOW CONTROL

The Underflow Control is used to indicate the occurrence of a calculation underflow condition within an Entity *since the last Get Underflow request*. Calculation underflow occurs when an attempt is made to assign a negative value to an unsigned variable. If implemented, this Control shall be Read-Only. Responding to the Get request returns the CUR attribute, and then clears its value. An Underflow Control shall have only the CUR attribute. The value of an Underflow Control CUR attribute shall be either TRUE (underflow condition occurred) or FALSE (normal).

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

The parameter block for this Control request uses Layout 1 (See Section 5.2.1.3.1, “Layout 1 Parameter Block.”)

5.2.1.4.3 OVERFLOW CONTROL

The Overflow Control is used to indicate the occurrence of a calculation overflow condition within an Entity *since the last Get Overflow request*. Calculation overflow occurs when a value is too positive or too negative to be represented after a signed calculation and when it is too positive after an unsigned calculation. If implemented, this Control shall be Read-Only. Responding to the Get request returns the CUR attribute, and then clears its value. An Overflow Control shall have only the CUR attribute. The value of an Overflow Control CUR attribute shall be either TRUE (overflow condition occurred) or FALSE (normal).

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

The parameter block for this Control request uses Layout 1 (See Section 5.2.1.3.1, “Layout 1 Parameter Block.”)

5.2.1.4.4 POWER DOMAIN CONTROL

This Control shall be supported as Read-Write whenever the Audio Function advertises the existence of at least one Power Domain through the presence of at least one Power Domain descriptor. (It is therefore not necessary to advertise the existence of the Control via the regular **bmControls** mechanism.) Otherwise, it shall not be supported.

The Power Domain Control is used to selectively bring parts of the Audio Function (a Power Domain) into different Power Domain States. The Power Domain Control is effectively an array of individual Controls, one for

each defined Power Domain. There shall be a Control present for each Power Domain and each Control shall support the CUR attribute. The RANGE(MIN, MAX, RES) attributes shall not be supported. The CUR attribute can only have positive values ranging from 0 (0x00) to 2 (0x02), indicating Power Domain State D0 to D2.

The Control Selector field shall be set to AC_POWER_DOMAIN_CONTROL and the Channel Number field shall be set to zero (master Control). The Entity ID field shall be set to the Power Domain ID of the desired Power Domain.

The parameter block for this Control request uses Layout 1 (See Section 5.2.1.3.1, “Layout 1 Parameter Block.”)

5.2.1.5 INTERFACE CONTROL REQUESTS

These requests are used to manipulate the Controls inside an AudioControl interface of the Audio Function. The exact layout of the request is defined in Section 5.2.1.3, “Control Request Parameter Block Layout.”

The following paragraphs present a detailed description of all possible Controls an AudioControl interface can incorporate. For each Control, the supported attributes and their value ranges are specified. Also, the appropriate Control Selector value and the layout type of the parameter blocks are listed. The Control Selector codes are defined in Appendix A.23.1, “AudioControl Interface Control Selectors.”

5.2.1.5.1 LATENCY CONTROL

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

Note: The presence of the Latency Controls is advertised in the class-specific AudioControl Interface descriptor and not repeated in every Terminal and Unit descriptor. Its functionality is described here, although the AudioControl interface by itself does not contain a Latency Control.

The Latency Control is used to accurately report the latency, expressed in nanoseconds, incurred by the addressed Entity. If implemented, this Control shall be Read-Only. A Latency Control shall have only the CUR attribute. The settings for the CUR attribute can range from 0 ns (0x00000000) to 4,294,967,295 ns (0xFFFFFFFF) in steps of 1 ns (0x00000001).

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

The parameter block for this Control request uses Layout 3 (See Section 5.2.1.3.3, “Layout 3 Parameter Block.”)

5.2.1.6 TERMINAL CONTROL REQUEST

This request is used to manipulate the Controls inside a Terminal of the Audio Function. The exact layout of the request is defined in Section 5.2.1.3, “Control Request Parameter Block Layout.”

The following paragraphs present a detailed description of all possible Controls a Terminal can incorporate. For each Control, the supported attributes and their value ranges are specified. Also, the appropriate Control Selector value and the layout type of the parameter blocks are listed. The Control Selector codes are defined in Appendix A.23.5, “Terminal Control Selectors.”

5.2.1.6.1 INSERTION CONTROL

The Insertion Control is used to examine the insertion state of Connectors that are associated with the Terminal. If implemented, this Control shall be Read-Only. An Insertion Control shall have only the CUR

attribute. The CUR attribute returns the **bmaConInserted** field that contains a bitmap where each bit represents the insertion state of one of the Connectors, associated with the Terminal. For this purpose, Connectors are identified by their ordinal number. If bit D_{i-1} is set, then Connector(i) is inserted. If bit D_{i-1} is reset, then Connector(i) is not inserted. Note that bit D_{i-1} only has meaning when bit D2 in the corresponding **bmaConAttributes(i)** field of the Connectors descriptor is set, indicating that Connector(i) is able to report meaningful insertion/removal states. For all Connectors, associated with the Terminal, that are not able to report insertion/removal state, the corresponding bit in the returned **bmaConInserted** bitmap shall always be set to zero. Likewise, all unused padding bits in the **bmConInserted** bitmap shall always be set to zero. The **bSize** field indicates the length in bytes of the **bmConInserted** field.

Note: The Audio Function is responsible to perform insertion detection and perform proper de-bounce so that only stable insertion and removal events are reported to the Host.

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

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

Table 5-9: Insertion Control CUR Parameter Block

wLength		1+n		
Offset	Field	Size	Value	Description
0	bSize	1	Number	Indicates the number of bytes in the bmConInserted bitmap: n
1	bmConInserted	n	Bitmap	Each set bit represents a currently inserted connector.

5.2.1.6.2 OVERLOAD CONTROL

The Overload Control is used to indicate the existence of an overload condition within a Terminal. (E.g. overvoltage at the analog line-in connector; thermal overload in powered speakers; etc.) If implemented, this Control shall be Read-Only. An Overload Control shall have only the CUR attribute. The value of an Overload Control CUR attribute shall be either TRUE (overload condition exists) or FALSE (normal).

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

The parameter block for this Control request uses Layout 1 (See Section 5.2.1.3.1, “Layout 1 Parameter Block.”)

5.2.1.6.3 UNDERFLOW CONTROL

See Section 5.2.1.4.2, “Underflow Control” for a detailed description. XX shall be replaced by TE.

5.2.1.6.4 OVERFLOW CONTROL

See Section 5.2.1.4.3, “Overflow Control” for a detailed description. XX shall be replaced by TE.

5.2.1.7 MIXER UNIT CONTROL REQUEST

This request is used to manipulate the Controls inside a Mixer Unit of the Audio Function. The exact layout of the request is defined in Section 5.2.1.3, “Control Request Parameter Block Layout.”

The **wValue** field specifies the Control Selector (CS) in the high byte and the Mixer Control Number (MCN) or the Channel Number (CN) in the low byte. If the CS value indicates that a Mixer Control is addressed (CS = MIXER_CONTROL), then the low byte contains the MCN. The MCN is derived according to the rules established in Section 4.5.2.5, “Mixer Unit Descriptor.”

The following paragraphs present a detailed description of all possible Controls a Mixer Unit can incorporate. For each Control, the supported attributes and their value ranges are specified. Also, the appropriate Control Selector value and the layout type of the parameter blocks are listed. The Control Selector codes are defined in Appendix A.23.6, “Mixer Control Selectors.”

5.2.1.7.1 MIXER CONTROL

A Mixer Unit consists of a number of N Mixer Controls, either programmable or fixed. At a minimum, all N Mixer Controls shall be present and implemented as Read-Only and support the CUR attribute. If a Mixer Control is implemented as Read-Write (programmable), the RANGE(MIN, MAX, RES) attributes shall also be supported. The settings for the CUR, MIN, and MAX attributes can range from +127.9961 dB (0x7FFF) down to -127.9961 dB (0x8001) in steps of 1/256 dB or 0.00390625 dB (0x0001). The settings for the RES attribute can only have positive values and range from 1/256 dB (0x0001) to +127.9961 dB (0x7FFF).

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

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

The parameter block for this Control request uses Layout 2 (See Section 5.2.1.3.2, “Layout 2 Parameter Block.”)

5.2.1.7.2 UNDERFLOW CONTROL

See Section 5.2.1.4.2, “Underflow Control” for a detailed description. XX shall be replaced by MU.

5.2.1.7.3 OVERFLOW CONTROL

See Section 5.2.1.4.3, “Overflow Control” for a detailed description. XX shall be replaced by MU.

5.2.1.8 SELECTOR UNIT CONTROL REQUEST

This request is used to manipulate the Controls inside a Selector Unit of the Audio Function. The exact layout of the request is defined in Section 5.2.1.3, “Control Request Parameter Block Layout.”

The following paragraphs present a detailed description of all possible Controls a Selector Unit can incorporate. For each Control, the supported attributes and their value ranges are specified. Also, the appropriate Control Selector value and the layout type of the parameter blocks are listed. The Control Selector codes are defined in Appendix A.23.7, “Selector Control Selectors.”

5.2.1.8.1 SELECTOR CONTROL

A Selector Unit represents a multi-channel source selector, capable of selecting among a number of identically configured Clusters. The Selector Control shall be present and at least be implemented as Read-Only. (Means external to USB may change this Control.) A Selector Control shall have only the CUR attribute. The valid range for the CUR attribute is from one up to the number of Input Pins of the Selector Unit. This value can be found in the **bNrInPins** field of the Selector Unit descriptor.

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

The parameter block for this Control request uses Layout 1 (See Section 5.2.1.3.1, “Layout 1 Parameter Block.”)

5.2.1.9 FEATURE UNIT CONTROL REQUEST

This request is used to manipulate the Controls inside a Feature Unit of the Audio Function. The exact layout of the request is defined in Section 5.2.1.3, “Control Request Parameter Block Layout.”

The following paragraphs present a detailed description of all possible Controls a Feature Unit can incorporate. For each Control, the supported attributes and their value ranges are specified. Also, the appropriate Control Selector value and the layout type of the parameter blocks are listed. The Control Selector codes are defined in Appendix A.23.8, “Feature Unit Control Selectors.”

5.2.1.9.1 MUTE CONTROL

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

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

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

The parameter block for this Control request uses Layout 1 (See Section 5.2.1.3.1, “Layout 1 Parameter Block.”)

5.2.1.9.2 VOLUME CONTROL

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

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

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

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

The parameter block for this Control request uses Layout 2 (See Section 5.2.1.3.2, “Layout 2 Parameter Block.”)

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

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

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

The parameter block for this Control request uses Layout 1 (See Section 5.2.1.3.1, “Layout 1 Parameter Block.”)

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

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

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

The parameter block for this Control request uses Layout 1 (See Section 5.2.1.3.1, “Layout 1 Parameter Block.”)

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

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

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

The parameter block for this Control request uses Layout 1 (See Section 5.2.1.3.1, “Layout 1 Parameter Block.”)

5.2.1.9.6 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 5-10: Band Numbers and Center Frequencies (ANSI S1.11-1986 Standard)

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

Note: Bands marked with an asterisk (*) are those present in an octave equalizer.

A Feature Unit that supports the Graphic Equalizer Control is not required to implement the full set of filters. A subset (for example, octave bands) may be implemented. During a 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 shall be in ascending order. If the number of bits set in the **bmBandsPresent** field does not match the number of parameters specified in the following block, the control pipe shall indicate a stall.

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

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

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

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

Table 5-11: Graphic Equalizer Control CUR Parameter Block

wLength		4+(number of bits set in bmBandsPresent : NrBits)		
Offset	Field	Size	Value	Description
0	bmBandsPresent	4	Bitmap	A bit set indicates the band is present: D0: Band 14 is present. D1: Band 15 is present. ... D29: Band 43 is present. D30: Reserved. D31: Reserved.
4	bCUR(Lowest)	1	Number	The setting for the CUR attribute of the lowest band present: 0x7F: +31.75 dB. 0x7E: +31.50 dB. ... 0x00: 0.00 dB. ... 0x82: -31.50 dB. 0x81: -31.75 dB. 0x80: -32.00 dB.
...	...	...	...	...
4+(NrBits-1)	bCUR(Highest)	1	Number	The setting for the CUR attribute of the highest band present.

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

Table 5-12: Graphic Equalizer Control RANGE Parameter Block

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

5.2.1.9.7 AUTOMATIC GAIN CONTROL

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

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

The Control Selector field shall be set to FU_AUTOMATIC_GAIN_CONTROL and the Channel Number field indicates the desired Channel.

The parameter block for this Control request uses Layout 1 (See Section 5.2.1.3.1, "Layout 1 Parameter Block.")

5.2.1.9.8 DELAY CONTROL

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

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

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

The parameter block for this Control request uses Layout 3 (See Section 5.2.1.3.3, "Layout 3 Parameter Block.")

5.2.1.9.9 BASS BOOST CONTROL

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

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

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

The parameter block for this Control request uses Layout 1 (See Section 5.2.1.3.1, "Layout 1 Parameter Block.")

5.2.1.9.10 LOUDNESS CONTROL

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

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

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

The parameter block for this Control request uses Layout 1 (See Section 5.2.1.3.1, "Layout 1 Parameter Block.")

5.2.1.9.11 INPUT GAIN CONTROL

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

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

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

The parameter block for this Control request uses Layout 2 (See Section 5.2.1.3.2, "Layout 2 Parameter Block.")

5.2.1.9.12 INPUT GAIN PAD CONTROL

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

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

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

The parameter block for this Control request uses Layout 2 (See Section 5.2.1.3.2, "Layout 2 Parameter Block.")

5.2.1.9.13 PHASE INVERTER CONTROL

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

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

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

The parameter block for this Control request uses Layout 1 (See Section 5.2.1.3.1, "Layout 1 Parameter Block.")

5.2.1.9.14 UNDERFLOW CONTROL

See Section 5.2.1.4.2, "Underflow Control" for a detailed description. XX shall be replaced by FU.

5.2.1.9.15 OVERFLOW CONTROL

See Section 5.2.1.4.3, "Overflow Control" for a detailed description. XX shall be replaced by FU.

5.2.1.10 EFFECT UNIT CONTROL REQUEST

This request is used to manipulate the Controls inside an Effect Unit of the Audio Function. The exact layout of the request is defined in Section 5.2.1.3, “Control Request Parameter Block Layout.”

The following paragraphs present a detailed description of all possible Controls an Effect Unit can incorporate. For each Control, the supported attributes and their value ranges are specified. Also, the appropriate Control Selector value and the layout type of the parameter blocks are listed. The Control Selector codes are defined in Appendix A.23.9, “Effect Unit Control Selectors.”

5.2.1.10.1 PARAMETRIC EQUALIZER SECTION EFFECT UNIT

5.2.1.10.1.1 ENABLE CONTROL

This Control shall be present and implemented as Read-Write. See Section 5.2.1.4.1, “Enable Control” for a detailed description. XX shall be replaced by PE.

5.2.1.10.1.2 CENTER FREQUENCY CONTROL

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

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

Note: A discrete list of supported center frequencies can be expressed using the method as explained in Section 5.2.1.1, “Control Attributes.”

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

The parameter block for this Control request uses Layout 3 (See Section 5.2.1.3.3, “Layout 3 Parameter Block.”)

5.2.1.10.1.3 QFACTOR CONTROL

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

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

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

The parameter block for this Control request uses Layout 3 (See Section 5.2.1.3.3, “Layout 3 Parameter Block.”)

Note: The Q-factor of a filter is defined as the ratio of the center frequency to the bandwidth measured at the -3 dB point. The result of a Q setting of 10 for a filter set to 1000 Hz is a bandwidth of 100 Hz. Likewise, a center frequency of 5325 Hz and a Q setting of 7.25 results in a bandwidth of 734.48275862 Hz.

5.2.1.10.1.4 GAIN CONTROL

The Gain Control is used to manipulate the amount of gain or attenuation at the center frequency of the PEQS Effect Unit. A Gain Control shall support the CUR and RANGE(MIN, MAX, RES) attributes. The settings for the CUR, MIN, and MAX attributes can range from +127.9961 dB (0x7FFF) down to -127.9961 dB (0x8001) in steps

of 1/256 dB or 0.00390625 dB (0x0001). The settings for the RES attribute can only have positive values and range from 1/256 dB (0x0001) to +127.9961 dB (0x7FFF).

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

The parameter block for this Control request uses Layout 2 (See Section 5.2.1.3.2, “Layout 2 Parameter Block.”)

5.2.1.10.1.5 UNDERFLOW CONTROL

See Section 5.2.1.4.2, “Underflow Control” for a detailed description. XX shall be replaced by PE.

5.2.1.10.1.6 OVERFLOW CONTROL

See Section 5.2.1.4.3, “Overflow Control” for a detailed description. XX shall be replaced by PE.

5.2.1.10.2 REVERBERATION EFFECT UNIT

5.2.1.10.2.1 ENABLE CONTROL

This Control shall be present and implemented as Read-Write. See Section 5.2.1.4.1, “Enable Control” for a detailed description. XX shall be replaced by RV.

5.2.1.10.2.2 TYPE CONTROL

The Type Control is a macro parameter that allows global settings of reverb parameters within the Reverberation Effect Unit. When a certain reverb type is selected, each reverb parameter will be set to the most suitable value. The Type Control shall support the CUR and RANGE(MIN, MAX, RES) attributes. The valid range for the CUR, MIN, and MAX attributes is from 0 to 255. The RES attribute can only have a value of 1.

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

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

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

The parameter block for this Control request uses Layout 1 (See Section 5.2.1.3.1, “Layout 1 Parameter Block.”)

5.2.1.10.2.3 LEVEL CONTROL

The Level Control is used to set the amount of reverberant sound introduced by the Reverberation Effect Unit. The Level Control shall support the CUR and RANGE(MIN, MAX, RES) attributes. The valid range for the CUR, MIN, MAX, and RES attributes is from 0 % to 255 %, compared to the level of the original signal.

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

The parameter block for this Control request uses Layout 1 (See Section 5.2.1.3.1, “Layout 1 Parameter Block.”)

5.2.1.10.2.4 TIME CONTROL

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

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

The parameter block for this Control request uses Layout 2 (See Section 5.2.1.3.2, “Layout 2 Parameter Block.”)

5.2.1.10.2.5 DELAY FEEDBACK CONTROL

The Delay Feedback Control is used when the reverb type is set to Type 6 (Delay) or Type 7 (Panning Delay). It sets the way in which delay repeats. The Delay Feedback Control range shall support the CUR and RANGE(MIN, MAX, RES) attributes. The valid range for the CUR, MIN, MAX, and RES attributes is from 0 % to 255 %. Higher values result in more delay repeats.

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

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

The parameter block for this Control request uses Layout 1 (See Section 5.2.1.3.1, “Layout 1 Parameter Block.”)

5.2.1.10.2.6 PRE-DELAY CONTROL

The Pre-Delay Control is used to set the delay time between the original source and the initial reflection in the reverberation, introduced by the Reverberation Effect Unit. The Pre-Delay Control shall support the CUR and RANGE(MIN, MAX, RES) attributes. The settings for the CUR, MIN, MAX, and RES attributes can range from 0 ms (0x0000) to 65535 ms (0xFFFF) in steps of 1 ms (0x0001).

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

The parameter block for this Control request uses Layout 2 (See Section 5.2.1.3.2, “Layout 2 Parameter Block.”)

5.2.1.10.2.7 DENSITY CONTROL

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

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

The parameter block for this Control request uses Layout 1 (See Section 5.2.1.3.1, “Layout 1 Parameter Block.”)

5.2.1.10.2.8 HI-FREQ ROLL-OFF CONTROL

The Hi-Freq Roll-Off Control is used to set the frequency of a low pass filter on the reverberant sound introduced by the Reverberation Effect Unit. The Hi-Freq Roll-Off Control shall support the CUR and RANGE(MIN, MAX, RES) attributes. The settings for the CUR, MIN, MAX and RES attributes can range from 0 Hz (0x00000000) to 4,294,967,295 Hz (0xFFFFFFFF) in steps of 1 Hz (0x00000001).

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

The parameter block for this Control request uses Layout 3 (See Section 5.2.1.3.3, “Layout 3 Parameter Block.”)

5.2.1.10.2.9 UNDERFLOW CONTROL

See Section 5.2.1.4.2, “Underflow Control” for a detailed description. XX shall be replaced by RV.

5.2.1.10.2.10 OVERFLOW CONTROL

See Section 5.2.1.4.3, “Overflow Control” for a detailed description. XX shall be replaced by RV.

5.2.1.10.3 MODULATION DELAY EFFECT UNIT

5.2.1.10.3.1 ENABLE CONTROL

See Section 5.2.1.4.1, “Enable Control” for a detailed description. XX shall be replaced by MD.

5.2.1.10.3.2 BALANCE CONTROL

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

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

The parameter block for this Control request uses Layout 1 (See Section 5.2.1.3.1, “Layout 1 Parameter Block.”)

5.2.1.10.3.3 RATE CONTROL

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

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

The parameter block for this Control request uses Layout 2 (See Section 5.2.1.3.2, “Layout 2 Parameter Block.”)

5.2.1.10.3.4 DEPTH CONTROL

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

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

The parameter block for this Control request uses Layout 2 (See Section 5.2.1.3.2, “Layout 2 Parameter Block.”)

5.2.1.10.3.5 TIME CONTROL

The Time Control is used to set the length of the delay time introduced by the Modulation Delay Effect Unit. Higher values result in longer times. The Time Control shall support the CUR and RANGE(MIN, MAX, RES)

attributes. The settings for the CUR, MIN, MAX, and RES attributes can range from 0 ms (0x0000) to 255.9961 ms (0xFFFF) in steps of 1/256 ms or 0.00390625 ms (0x0001).

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

The parameter block for this Control request uses Layout 2 (See Section 5.2.1.3.2, “Layout 2 Parameter Block.”)

5.2.1.10.3.6 FEEDBACK LEVEL CONTROL

The Feedback Level Control is used to set the level at which the effected (delayed) sound introduced by the Modulation Delay Effect Unit is mixed back into to its own input. Higher values result in higher level of feedback. The Feedback Level Control range shall support the CUR and RANGE(MIN, MAX, RES) attributes. The valid range for the CUR, MIN, MAX, and RES attributes is from 0 % to 255 %. Higher values result in more delay repeats.

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

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

The parameter block for this Control request uses Layout 1 (See Section 5.2.1.3.1, “Layout 1 Parameter Block.”)

5.2.1.10.3.7 UNDERFLOW CONTROL

See Section 5.2.1.4.2, “Underflow Control” for a detailed description. XX shall be replaced by MD.

5.2.1.10.3.8 OVERFLOW CONTROL

See Section 5.2.1.4.3, “Overflow Control” for a detailed description. XX shall be replaced by MD.

5.2.1.10.4 DYNAMIC RANGE COMPRESSOR EFFECT UNIT

5.2.1.10.4.1 ENABLE CONTROL

See Section 5.2.1.4.1, “Enable Control” for a detailed description. XX shall be replaced by DR.

5.2.1.10.4.2 COMPRESSION RATIO CONTROL

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

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

The parameter block for this Control request uses Layout 2 (See Section 5.2.1.3.2, “Layout 2 Parameter Block.”)

5.2.1.10.4.3 MAXAMPL CONTROL

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

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

The parameter block for this Control request uses Layout 2 (See Section 5.2.1.3.2, “Layout 2 Parameter Block.”)

5.2.1.10.4.4 THRESHOLD CONTROL

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

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

The parameter block for this Control request uses Layout 2 (See Section 5.2.1.3.2, “Layout 2 Parameter Block.”)

5.2.1.10.4.5 ATTACK TIME CONTROL

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

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

The parameter block for this Control request uses Layout 2 (See Section 5.2.1.3.2, “Layout 2 Parameter Block.”)

5.2.1.10.4.6 RELEASE TIME CONTROL

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

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

The parameter block for this Control request uses Layout 2 (See Section 5.2.1.3.2, “Layout 2 Parameter Block.”)

5.2.1.10.4.7 UNDERFLOW CONTROL

See Section 5.2.1.4.2, “Underflow Control” for a detailed description. XX shall be replaced by DR.

5.2.1.10.4.8 OVERFLOW CONTROL

See Section 5.2.1.4.3, “Overflow Control” for a detailed description. XX shall be replaced by DR.

5.2.1.11 PROCESSING UNIT CONTROL REQUEST

This request is used to manipulate the Controls inside a Processing Unit of the Audio Function. The exact layout of the request is defined in Section 5.2.1.3, “Control Request Parameter Block Layout.”

The following paragraphs present a detailed description of all possible Controls a Processing Unit can incorporate. For each Control, the supported attributes and their value ranges are specified. Also, the

appropriate Control Selector value and the layout type of the parameter blocks are listed. The Control Selector codes are defined in Appendix A.23.10, “Processing Unit Control Selectors.”

5.2.1.11.1 UP/DOWN-MIX PROCESSING UNIT

5.2.1.11.1.1 MODE SELECT CONTROL

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

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

The parameter block for this Control request uses Layout 1 (See Section 5.2.1.3.1, “Layout 1 Parameter Block.”)

5.2.1.11.1.2 UNDERFLOW CONTROL

See Section 5.2.1.4.2, “Underflow Control” for a detailed description. XX shall be replaced by UD.

5.2.1.11.1.3 OVERFLOW CONTROL

See Section 5.2.1.4.3, “Overflow Control” for a detailed description. XX shall be replaced by UD.

5.2.1.11.2 STEREO EXTENDER PROCESSING UNIT

5.2.1.11.2.1 WIDTH CONTROL

The Width Control is used to change the spatial appearance of the stereo image, produced by the Stereo Extender Processing Unit. The Width Control shall support the CUR and RANGE(MIN, MAX, RES) attributes. The valid range for the CUR, MIN, MAX, and RES attributes is from 0 to 255.

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

The parameter block for this Control request uses Layout 1 (See Section 5.2.1.3.1, “Layout 1 Parameter Block.”)

5.2.1.11.2.2 UNDERFLOW CONTROL

See Section 5.2.1.4.2, “Underflow Control” for a detailed description. XX shall be replaced by ST.

5.2.1.11.2.3 OVERFLOW CONTROL

See Section 5.2.1.4.3, “Overflow Control” for a detailed description. XX shall be replaced by ST.

5.2.1.12 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 shall or can support.

This request is used to manipulate the Controls inside an Extension Unit of the Audio Function. The exact layout of the request is defined in Section 5.2.1.3, “Control Request Parameter Block Layout.”

The following paragraphs present a detailed description of all possible Controls a Processing Unit can incorporate. For each Control, the supported attributes and their value ranges are specified. Also, the

appropriate Control Selector value and the layout type of the parameter blocks are listed. Issuing non-supported Control Selectors to an Extension Unit leads to a control pipe stall. The Control Selector codes are defined in Appendix A.23.11, “Extension Unit Control Selectors.”

5.2.1.12.1.1 UNDERFLOW CONTROL

See Section 5.2.1.4.2, “Underflow Control” for a detailed description. XX shall be replaced by XU.

5.2.1.12.1.2 OVERFLOW CONTROL

See Section 5.2.1.4.3, “Overflow Control” for a detailed description. XX shall be replaced by XU.

5.2.1.13 CLOCK SOURCE CONTROL REQUEST

This request is used to manipulate the Controls inside a Clock Source Entity of the Audio Function. The exact layout of the request is defined in Section 5.2.1.3, “Control Request Parameter Block Layout.”

The following paragraphs present a detailed description of all possible Controls a Clock Source Entity can incorporate. For each Control, the supported attributes and their value ranges are specified. Also, the appropriate Control Selector value and the layout type of the parameter blocks are listed. The Control Selector codes are defined in Appendix A.23.2, “Clock Source Control Selectors.”

5.2.1.13.1 SAMPLING FREQUENCY CONTROL

The Sampling Frequency Control is used to manipulate the actual sampling frequency of the clock signal that is generated by the Clock Source Entity. At a minimum, this Control shall be supported as Read-Only by every Clock Source Entity.

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

Note: A discrete list of supported sampling frequencies can be expressed using the method as explained in Section 5.2.1.1, “Control Attributes.”

In many cases, the Clock Source Entity represents a crystal oscillator based generator with a single fixed frequency. In that case, the Set request is not supported. Additionally, the Set request may not be supported if the Clock Source entity represents an external clock which cannot be controlled by the Audio Function hardware.

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

The parameter block for this Control request uses Layout 3 (See Section 5.2.1.3.3, “Layout 3 Parameter Block.”)

5.2.1.13.2 CLOCK VALIDITY CONTROL

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

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

The parameter block for this Control request uses Layout 1 (See Section 5.2.1.3.1, “Layout 1 Parameter Block.”)

5.2.1.14 CLOCK SELECTOR CONTROL REQUEST

This request is used to manipulate the Controls inside a Clock Selector Entity of the Audio Function. The exact layout of the request is defined in Section 5.2.1.3, “Control Request Parameter Block Layout.”

The following paragraphs present a detailed description of all possible Controls a Clock Selector Entity can incorporate. For each Control, the supported attributes and their value ranges are specified. Also, the appropriate Control Selector value and the layout type of the parameter blocks are listed. The Control Selector codes are defined in Appendix A.23.3, “Clock Selector Control Selectors.”

5.2.1.14.1 CLOCK SELECTOR CONTROL

A Clock Selector Entity represents a multi-input source selector, capable of selecting among a number of clock signals. The Clock Selector Control shall be present and at least be implemented as Read-Only. (Means external to USB may change this Control.) The valid range for the CUR attribute is from one up to the number of Clock Input Pins of the Clock Selector. This value can be found in the **bNrInPins** field of the Clock Selector descriptor.

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

The parameter block for this Control request uses Layout 1 (See Section 5.2.1.3.1, “Layout 1 Parameter Block.”)

5.2.1.15 CLOCK MULTIPLIER CONTROL REQUEST

This request is used to manipulate the Controls inside a Clock Multiplier Entity of the Audio Function. The exact layout of the request is defined in Section 5.2.1.3, “Control Request Parameter Block Layout.”

The following paragraphs present a detailed description of all possible Controls a Clock Multiplier Entity can incorporate. For each Control, the supported attributes and their value ranges are specified. Also, the appropriate Control Selector value and the layout type of the parameter blocks are listed. The Control Selector codes are defined in Appendix A.23.4, “Clock Multiplier Control Selectors.”

5.2.1.15.1 NUMERATOR CONTROL

At a minimum, the Numerator Control shall be supported as Read-Only. It is used to manipulate the factor P by which the incoming sampling clock signal is multiplied. This Control shall support the CUR and RANGE(MIN, MAX, RES) attributes. The settings for the CUR, MIN, MAX and RES attributes can range from 1 (0x0001) to $2^{16}-1$ (0xFFFF).

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

The parameter block for this Control request uses Layout 2 (See Section 5.2.1.3.2, “Layout 2 Parameter Block.”)

5.2.1.15.2 DENOMINATOR CONTROL

At a minimum, the Denominator Control shall be supported as Read-Only. It is used to manipulate the factor Q by which the incoming sampling clock signal is divided. This Control shall support the CUR and RANGE(MIN, MAX, RES) attributes. The settings for the CUR, MIN, MAX and RES attributes can range from 1 (0x0001) to $2^{16}-1$ (0xFFFF).

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

The parameter block for this Control request uses Layout 2 (See Section 5.2.1.3.2, “Layout 2 Parameter Block.”)

5.2.2 AUDIOSTREAMING REQUESTS

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

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

5.2.2.1 INTERFACE CONTROL REQUESTS

These requests are used to manipulate the Controls inside an AudioStreaming interface of the Audio Function. The exact layout of the request is defined in Section 5.2.1.3, “Control Request Parameter Block Layout.”

The following paragraphs present a detailed description of all possible Controls an AudioStreaming interface can incorporate. For each Control, the supported attributes and their value ranges are specified. Also, the appropriate Control Selector value and the layout type of the parameter blocks are listed. The Control Selector codes are defined in Appendix A.23.12, “AudioStreaming Interface Control Selectors.”

5.2.2.1.1 ACTIVE ALTERNATE SETTING CONTROL

This Control is used to inform Host software which Alternate Setting of an AudioStreaming interface is currently active. The main purpose of this Control is to notify the Host (through an interrupt) that the last selected Alternate Setting is no longer valid. There can be a variety of reasons why this could happen. For instance, increasing the sampling frequency of a Clock Source Entity might render the current Alternate Setting of an interface connected to that Clock Source invalid because more bandwidth is now needed than is available in the current Alternate Setting.

This specification does not allow an interface to change from one active Alternate Setting to another without Host intervention. Whenever an Alternate Setting becomes invalid, the interface is required to switch to (idle) Alternate Setting zero. If this situation may occur in the Audio Function, this Control (and the Valid Alternate Settings Control) shall be present and implemented as Read-Only. It always provides the currently active Alternate Setting for the interface. The Host software needs to then take appropriate action to reactivate the interface by switching to a valid Alternate Setting. An Active Alternate Setting Control shall have only the CUR attribute. The value of an Active Alternate Setting Control CUR attribute shall only be either the last set Alternate Setting or zero.

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

The parameter block for this Control request uses Layout 1 (See Section 5.2.1.3.1, “Layout 1 Parameter Block.”)

5.2.2.1.2 VALID ALTERNATE SETTINGS CONTROL

This Control is used to inform Host software what the currently possible valid Alternate Settings are for an AudioStreaming interface. If the Active Alternate Setting Control is present, then this Control shall also be present and be implemented as Read-Only. It always provides a list of currently valid Alternate Settings for the interface. A Valid Alternate Settings Control shall have only the CUR attribute. The value of a Valid Alternate Setting Control CUR attribute is a bitmap, returned in the **bmValidAltSettings** field, that contains a bit for each possible active Alternate Setting. The **bSize** field indicates the length in bytes of the **bmValidAltSettings** field.

A bit set means that this Alternate Setting is currently valid. A bit cleared means that this Alternate Setting is currently not valid. Bit D0 corresponds to Alternate Setting 0 and shall always be set since it is always a possible valid setting. Bit D1 corresponds to Alternate Setting 1. Bit Dm corresponds to Alternate Setting m. All bits that do not correspond to an existing Alternate Setting shall be set to 0. An attempt to set the interface to an invalid Alternate Setting (through the standard Set Interface request) will result in a control pipe stall.

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

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

Table 5-13: Valid Alternate Settings Control CUR Parameter Block

wLength		1+n		
Offset	Field	Size	Value	Description
0	bSize	1	Number	Indicates the number of bytes in the bmValidAltSettings bitmap: n
1	bmValidAltSettings	n	Bitmap	Each set bit represents a currently valid Alternate Setting for the interface.

5.2.2.1.3 AUDIO DATA FORMAT CONTROL

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

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

The parameter block for this Control request is as follows:

Table 5-14: Audio Data Format Control CUR Parameter Block

wLength		8		
Offset	Field	Size	Value	Description
0	bmFormats	8	Bitmap	The Audio Data Format(s) that can be used to communicate with this interface. See the <i>USB Audio Data Formats</i> document for further details.

5.2.2.2 ENDPOINT CONTROL REQUEST

This request is used to manipulate the Controls inside an AudioStreaming endpoint of the Audio Function. The exact layout of the request is defined in Section 5.2.1.3, “Control Request Parameter Block Layout.”

The following paragraphs present a detailed description of all possible Controls an Endpoint can incorporate. For each Control, the supported attributes and their value ranges are specified. Also, the appropriate Control Selector value and the layout type of the parameter blocks are listed. The Control Selector codes are defined in Appendix A.23.13, “Endpoint Control Selectors.”

5.2.2.2.1 PITCH CONTROL

The Pitch Control enables or disables the ability of an adaptive endpoint to dynamically track its sampling frequency. The Control is necessary because the clock recovery circuitry must be informed whether it should allow for relatively large swings in the sampling frequency. A Pitch Control shall have only the CUR attribute. The value of a Pitch Control CUR attribute shall be either TRUE or FALSE.

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

The parameter block for this Control request uses Layout 1 (See Section 5.2.1.3.1, “Layout 1 Parameter Block.”)

5.2.2.2.2 DATA OVERRUN CONTROL

The Data Overrun Control is used to indicate the occurrence of a data overrun (buffer overflow) condition within an Endpoint *since the last Get Data Overrun request*. If implemented, this Control shall be Read-Only. A Data Overrun Control shall have only the CUR attribute. The value of a Data Overrun Control CUR attribute shall be either TRUE (overrun condition occurred) or FALSE (normal).

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

The parameter block for this Control request uses Layout 1 (See Section 5.2.1.3.1, “Layout 1 Parameter Block.”)

5.2.2.2.3 DATA UNDERRUN CONTROL

The Data Underrun Control is used to indicate the occurrence of a data underrun (buffer underflow) condition within an Endpoint *since the last Get Data Underrun request*. If implemented, this Control shall be Read-Only. A Data Underrun Control shall have only the CUR attribute. The value of a Data Underrun Control CUR attribute shall be either TRUE (underrun condition occurred) or FALSE (normal).

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

The parameter block for this Control request uses Layout 1 (See Section 5.2.1.3.1, “Layout 1 Parameter Block.”)

5.2.3 ADDITIONAL REQUESTS

5.2.3.1 MEMORY REQUESTS

The Host can interact with an addressable Entity (Clock Entity, Terminal, Unit, Power domain, Interface or endpoint) within the Audio Function in a very generic way. The Entity presents a memory space to the Host whose layout depends on the implementation. The Memory request provides full access to this memory space.

This request is used to upload or download a parameter block into a particular Entity of the Audio Function.

Table 5-15: Memory Request Values

bmRequest Type	bRequest	wValue	wIndex	wLength	Data
00100001B 10100001B	MEM INTEN	Offset	Entity ID and Interface	Length of parameter block	Parameter block
00100010B 10100010B			Zero and Endpoint		

The **bRequest** field indicates that the MEM or INTEN attribute of the Entity is addressed.

The **wValue** field specifies a zero-based offset value that can be used to access only parts of the Entity’s memory space.

The **wIndex** field specifies the interface or endpoint to be addressed in the low byte and the Entity ID (Clock Entity ID, Unit ID, Terminal ID, or Power Domain 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 the **wIndex** field shall 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 shall indicate a stall.

The layout of the parameter block is implementation dependent. A device is required to reevaluate its memory space at the end of each Set Memory request.

The Interrupt Enable attribute is used to manipulate the memory space's ability to generate an interrupt when any of its memory locations change other than through Host manipulation. The Interrupt Enable attribute shall be supported for all Entity memory spaces that are able to generate an interrupt. When supported, the Interrupt Enable attribute is always Read-Write and therefore, both the Get and Set request shall be supported. The Interrupt Enable attribute can take only two values: 0 (False, Disabled) or 1 (True, Enabled). All Interrupt Enable attributes shall have a default value of 1 (Enabled).

5.2.3.2 CLASS-SPECIFIC STRING REQUEST

To overcome the limitations of the standard Get String descriptor request (limited to 255 device-wide strings), this specification provides a class-specific extension to retrieve a larger set of potentially larger String descriptors from the Audio Function.

The Get Class-specific String request is used to retrieve class-specific String descriptors and shall be supported if the Audio Function contains at least one class-specific String descriptor. These String descriptors are uniquely identified within the Audio Function through their **wStrDescrID** and **iLangID** values.

Table 5-16: String Request

bmRequest Type	bRequest	wValue	wIndex	wLength	Data
10100001B	STRING	wStrDescrID	iLangID and Interface	Length of parameter block	Parameter block

The **bmRequestType** field shall be set to 0b10100001 to indicate that this is a class-specific Get Request, directed to the AudioControl interface.

The **bRequest** field indicates that the class-specific set of String descriptors is targeted.

The **wValue** field specifies the String descriptor's **wStrDescrID**. For maximum interoperability between this class-specific method to retrieve String descriptors from the Audio Function and the standard Get String descriptor request, all Audio Function related standard String descriptors, including the LANGID code array at index 0, that can be retrieved using the standard Get String descriptor request shall also be retrievable using this class-specific String request by specifying zero in the high byte and the standard String descriptor index in the low byte of the **wStrDescrID** value. Whether other than Audio Function related standard String descriptors present in the device can be retrieved using this new method is implementation-dependent. Any newly defined class-specific String descriptor that uses the **wStrDescrID** value as an index, shall use **wStrDescrID** values starting from 256 onwards.

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

The values specified in the **wValue** and **wIndex** fields shall be appropriate to the recipient. Only existing **wStrDescrID** values in the Audio Function can be indexed and only appropriate **iLangID** and AudioControl interface numbers may be used. If the request specifies an unknown **wStrDescrID** or **iLangID** value or an unknown AudioControl interface number, the control pipe shall indicate a stall.

The length of the descriptor to return is indicated in the **wLength** field of the request. If the descriptor is longer than what is indicated in the **wLength** field, only the initial bytes of the descriptor are returned. If the

descriptor is shorter than what is indicated in the **wLength** field, the device indicates the end of the control transfer by sending a short packet when further data is requested.

The layout of the parameter block follows the class-specific String descriptor definition as outlined in Section 4.9, “Class-specific String descriptors.”

5.2.3.3 HIGH CAPABILITY DESCRIPTOR REQUEST

To overcome the limitations of the standard Get Descriptor request (limited to maximum 256 bytes long), this specification provides a class-specific method to retrieve larger descriptors from the Audio Function. Also, since High Capability descriptors have the ability to report changes dynamically, they can be used whenever there is a need for the descriptor to indicate that some of its values have changed (even when its length is less than 256 bytes).

The Get High Capability Descriptor request is used to retrieve High Capability descriptors and shall be supported whenever the Audio Function uses at least one High Capability descriptor. A High Capability descriptor is uniquely identified within an interface or endpoint by the **wDescriptorID** value.

Table 5-17: High Capability Descriptor Request

bmRequest Type	bRequest	wValue	wIndex	wLength	Data
10100001B	HIGH_CAPABILITY_DESCRIPTOR	wDescriptorID	Zero and Interface	Length of parameter block	Parameter block
10100010B			Zero and Endpoint		

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

The **bRequest** field indicates that a class-specific descriptor is targeted.

The **wValue** field specifies the descriptor’s **wDescriptorID** value.

When the request addresses an interface (**bmRequestType** = 10100001), the **wIndex** field specifies the interface in the low byte and zero in the high byte.

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

The values in the **wValue** and **wIndex** fields shall be appropriate to the recipient. Only existing **wDescriptorID** values can be used and only appropriate interface or endpoint numbers may be used. If the request specifies an unknown **wDescriptorID** value or an unknown interface or endpoint number, the control pipe shall indicate a stall.

The length of the descriptor to return is indicated in the **wLength** field of the request. If the descriptor is longer than what is indicated in the **wLength** field, only the initial bytes of the descriptor are returned. If the descriptor is shorter than what is indicated in the **wLength** field, the device indicates the end of the control transfer by sending a short packet when further data is requested.

The layout of the parameter block follows the High Capability descriptor definitions as outlined in Section 4.2, “Class-Specific Descriptors.”

6 INTERRUPTS

Interrupts are used as a means to inform the Host that a change has occurred in the current state of the Audio Function. This specification currently defines two different types of interrupts:

- **Memory Change:** Some internal Entity's memory location has been updated. Host software can be notified so that the appropriate action can be taken.
- **Control Change:** Some addressable Control inside the Audio Function changed one or more of its attribute values.

The Audio Controls inside a Clock Entity, Unit, Terminal or Power Domain can be the source of an interrupt. Likewise, any addressable Control inside the AudioControl interface or any of the AudioStreaming interfaces can generate an interrupt. Finally, all addressable Controls related to an audio endpoint can be the cause of an interrupt.

A change of state in the Audio Function is most often caused by a certain event that takes place. An event can either be user-initiated or device-initiated. User-initiated connector insertion or removal is a typical example of a user-initiated event. The Host could then switch selectors or mixers so as to play audio out of the just inserted device (e.g. a headphone) and stop playing audio out of the current device (e.g. a speaker set).

An example of a device-initiated event could be the following: An external device (e.g. an A/V receiver) could switch from PCM to AC-3 encoded data on its optical digital output, depending on the material that is currently being played. If this device is connected to the optical digital input of an Audio Function that has auto-detect capabilities, the interface on that Audio Function might need to be reconfigured (e.g. to start the AC-3 decoding process), maybe causing all other interfaces to change some aspect of their format, or even become unusable. The device could issue an interrupt, letting the Host know that the Audio Function needs reconfiguration.

6.1 INTERRUPT DATA MESSAGE

The actual type of interrupt (Memory Change, Control Change, Descriptor Change) and its originator is conveyed to the Host through the interrupt data message that is sent over the interrupt endpoint. It is then the responsibility of the Host to query the Audio Function for more detailed information about the cause of the interrupt through a Get Memory request or one of the Get Control or Get Descriptor requests as defined in Section 5.2, "Class-Specific Requests".

Note that if the Host directly changes the CUR attribute on any Control, that Control shall not generate an interrupt. Likewise, if the Host directly modifies a memory location within an Entity, that change shall not generate an interrupt.

Interrupts are considered to be of the 'edge-triggered' type, meaning that an interrupt is generated whenever an event occurs, but there is no specific action required from the Host to clear the interrupt condition. When the Host issues a Get request in response to the interrupt, the most current value of the addressed Control's attribute will be returned.

The interrupt data message is always 6 bytes in length. The first **blinfo** field is required for all interrupt data messages. It contains information in D0 indicating whether this is a vendor-specific interrupt (D0 = 0b1) or a class-specific interrupt (D0 = 0b0). Bit D1 indicates whether the interrupt originated from an interface (D1 = 0b0) or an endpoint (D1 = 0b1). Bits D7..2 of the **blinfo** field are reserved. For vendor-specific interrupts, the layout of the remainder of the interrupt message is undefined. For class-specific interrupts, the layout is defined as follows.

When the interrupt originates from an Entity in an interface (D1 = 0b0 in the **bInfo** field), the **wIndex** field specifies the interface in the low byte and the Entity ID (Clock Entity ID, Unit ID, Terminal ID, or Power Domain ID). For indicating the interface itself, an Entity ID of zero shall be specified in the high byte.

When the interrupt originates from an endpoint (D1 = 0b1 in the **bInfo** field), the **wIndex** field specifies the endpoint to be addressed in the low byte and zero in the high byte.

The **wValue** field interpretation is qualified by the value in the **wIndex** field. The layout of the **wValue** field changes depending on the addressed Entity. The **wValue** field follows exactly the same rules as outlined in Section 5, “Requests” for each of the supported Get Control requests. The **wValue** field returns the Control Selector (CS) in the high byte and the Channel Number (CN) in the low byte. The Control Selector and the Channel Number (CN) together indicate exactly which Control generated the interrupt. If a Control is channel-independent, then the Control is considered to be a master Control and the virtual channel zero is returned to indicate it (CN = 0).

There are two exceptions to the above. The first is when a Mixer Unit Control request returns CS = MU_MIXER_CONTROL in the high byte. Then the Mixer Control Number (MCN) is returned in the low byte. The second is the Memory request where the **wValue** field specifies a zero-based offset value that indicates the address of the location in the Entity’s memory space that generated the interrupt. If the offset value is zero, this indicates that multiple memory locations might have changed and the entire memory space needs to be examined.

The **bSourceType** field contains a constant, identifying the specific source type (attribute of the addressed Control or Entity, String, or High Capability descriptor) is causing the interrupt. Possible source types are:

- Current setting attribute (CUR)
- Range attribute (RANGE)
- Memory space attribute (MEM)
- String (STRING)
- High Capability Descriptor (HIGH_CAPABILITY_DESCRIPTOR)

When there are no interrupts pending, the interrupt endpoint shall NAK when polled.

The following table specifies the format of the interrupt message when D0 = 0b0:

Table 6-1: Interrupt Data Message Format

Offset	Field	Size	Value	Description
0	bInfo	1	Bitmap	D0: Vendor-specific. D1: Interface or Endpoint D7..2: Reserved.
1	bSourceType	1	Constant	The source type that caused the interrupt
2	wValue	2	Number	CS in the high byte and CN or MCN in the low byte for CUR and RANGE source type. Zero-based offset into memory space for MEM source type. wStrDescrID value for String source type. wDescriptorID value for High Capability Descriptor source type.
4	wIndex	2	Number	Entity ID or zero in the high byte and Interface or Endpoint in the low byte.

6.2 INTERRUPT SOURCES

Any Control, High Capability descriptor, class-specific String descriptor, or memory location within an addressable Entity of the Audio Function can be the source of an interrupt. The interrupt message contains enough information to determine exactly which Control or memory location caused the interrupt. The Host can then issue the normal Control or Memory requests to further qualify the cause of the interrupt.

IECNORM.COM : Click to view the full PDF of IEC 62680-1-5:2019

APPENDIX A. AUDIO DEVICE CLASS CODES

A.1 AUDIO FUNCTION CLASS CODE

Table A-1: Audio Function Class Code

Audio Function Class Code	Value
AUDIO_FUNCTION	AUDIO

A.2 AUDIO FUNCTION SUBCLASS CODES

Table A-2: Audio Function Subclass Codes

Audio Function Subclass Code	Value
FUNCTION_SUBCLASS_UNDEFINED	0x00
FULL_ADC_3_0	0x01
GENERIC_I/O	0x20
HEADPHONE	0x21
SPEAKER	0x22
MICROPHONE	0x23
HEADSET	0x24
HEADSET_ADAPTER	0x25
SPEAKERPHONE	0x26

A.3 AUDIO FUNCTION PROTOCOL CODES

Table A-3: Audio Function Protocol Codes

Audio Function Protocol Code	Value
FUNCTION_PROTOCOL_UNDEFINED	0x00
AF_VERSION_01_00	IP_VERSION_01_00
AF_VERSION_02_00	IP_VERSION_02_00
AF_VERSION_03_00	IP_VERSION_03_00

A.4 AUDIO INTERFACE CLASS CODE

Table A-4: Audio Interface Class Code

Audio Interface Class Code	Value
AUDIO	0x01

A.5 AUDIO INTERFACE SUBCLASS CODES

Table A-5: Audio Interface Subclass Codes

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

A.6 AUDIO INTERFACE PROTOCOL CODES

Table A-6: Audio Interface Protocol Codes

Audio Interface Protocol Code	Value
IP_VERSION_01_00	0x00
IP_VERSION_02_00	0x20
IP_VERSION_03_00	0x30

A.7 AUDIO FUNCTION CATEGORY CODES

Table A-7: Audio Function Category Codes

Audio Function Subclass Code	Value
FUNCTION_SUBCLASS_UNDEFINED	0x00
DESKTOP_SPEAKER	0x01
HOME_THEATER	0x02
MICROPHONE	0x03
HEADSET	0x04
TELEPHONE	0x05
CONVERTER	0x06
VOICE/SOUND_RECORDER	0x07
I/O_BOX	0x08
MUSICAL_INSTRUMENT	0x09
PRO-AUDIO	0x0A
AUDIO/VIDEO	0x0B
CONTROL_PANEL	0x0C
HEADPHONE	0x0D
GENERIC_SPEAKER	0x0E
HEADSET_ADAPTER	0x0F
SPEAKERPHONE	0x10
Reserved	0x11..0xFE
OTHER	0xFF

A.8 AUDIO CLASS-SPECIFIC DESCRIPTOR TYPES

Table A-8: Audio Class-specific Descriptor Types

Descriptor Type	Value
CS_UNDEFINED	0x20
CS_DEVICE	0x21
CS_CONFIGURATION	0x22
CS_STRING	0x23
CS_INTERFACE	0x24
CS_ENDPOINT	0x25
CS_CLUSTER	0x26

A.9 CLUSTER DESCRIPTOR SUBTYPES

Table A-9: Audio Class-Specific Cluster Descriptor Subtypes

Descriptor Subtype	Value
SUBTYPE_UNDEFINED	0x00

A.10 CLUSTER DESCRIPTOR SEGMENT TYPES

Table A-10: Cluster Descriptor Segment Types

Segment Type	Value
SEGMENT_UNDEFINED	0x00
CLUSTER_DESCRIPTION	0x01
CLUSTER_VENDOR_DEFINED	0x1F
CHANNEL_INFORMATION	0x20
CHANNEL_AMBISONIC	0x21
CHANNEL_DESCRIPTION	0x22
CHANNEL_VENDOR_DEFINED	0xFE
END_SEGMENT	0xFF

A.11 CHANNEL PURPOSE DEFINITIONS

Table A-11: Channel Purpose Definitions

Channel Purpose	Value
PURPOSE_UNDEFINED	0x00
GENERIC_AUDIO	0x01
VOICE	0x02
SPEECH	0x03
AMBIENT	0x04
REFERENCE	0x05
ULTRASONIC	0x06
VIBROKINETIC	0x07
NON_AUDIO	0xFF

A.12 CHANNEL RELATIONSHIP DEFINITIONS

Table A-12: Channel Relationship Definitions

USB Audio Channel Relationship		CEA-861.2 Channel Allocation	
Description	Acronym	Description	Acronym
RELATIONSHIP_UNDEFINED	UND	---	---
MONO	M	---	---
LEFT	L	---	---
RIGHT	R	---	---
ARRAY	AR	---	---
PATTERN_X	PX	---	---
PATTERN_Y	PY	---	---
PATTERN_A	PA	---	---
PATTERN_B	PB	---	---
PATTERN_M	PM	---	---
PATTERN_S	PS	---	---
Front Left	FL	Front Left	FL
Front Right	FR	Front Right	FR
Front Center	FC	Front Center	FC
Front Left of Center	FLC	Front Left of Center	FLc
Front Right of Center	FRC	Front Right of Center	FRc
Front Wide Left	FWL	Front left Wide	FLw
Front Wide Right	FWR	Front Right Wide	FRw
Side Left	SL	Side Left	SiL
Side Right	SR	Side Right	SiR
Surround Array Left	SAL	Left Surround	LS

USB Audio Channel Relationship		CEA-861.2 Channel Allocation	
Description	Acronym	Description	Acronym
Surround Array Right	SAR	Right Surround	RS
Back Left	BL	Back Left	BL
Back Right	BR	Back Right	BR
Back Center	BC	Back Center	BC
Back Left of Center	BLC	---	---
Back Right of Center	BRC	---	---
Back Wide Left	BWL	---	---
Back Wide Right	BWR	---	---
Top Center	TC	Top Center	TpC
Top Front Left	TFL	Top Front Left	TpFL
Top Front Right	TFR	Top Front Right	TpFR
Top Front Center	TFC	Top Front Center	TpFC
Top Front Left of Center	TFLC	---	---
Top Front Right of Center	TFRC	---	---
Top Front Wide Left	TFWL	---	---
Top Front Wide Right	TFWR	---	---
Top Side Left	TSL	Top Side Left	TpSiL
Top Side Right	TSR	Top Side Right	TpSiR
Top Surround Array Left	TSAL	Top Left Surround	TpLS
Top Surround Array Right	TSAR	Top Right Surround	TpRS
Top Back Left	TBL	Top Back Left	TpBL
Top Back Right	TBR	Top Back Right	TpBR
Top Back Center	TBC	Top Back Center	TpBC
Top Back Left Of Center	TBLC	---	---
Top Back Right Of Center	TBRC	---	---
Top Back Wide Left	TBWL	---	---
Top Back Wide Right	TBWR	---	---
Bottom Center	BC	---	---
Bottom Front Left	BFL	Bottom Front Left	BtFL
Bottom Front Right	BFR	Bottom Front Right	BtFR
Bottom Front Center	BFC	Bottom Front Center	BtFC
Bottom Front Left Of Center	BFLC	---	---
Bottom Front Right Of Center	BFRC	---	---
Bottom Front Wide Left	BFWL	---	---
Bottom Front Wide Right	BFWR	---	---
Bottom Side Left	BSL	---	---
Bottom Side Right	BSR	---	---

USB Audio Channel Relationship		CEA-861.2 Channel Allocation	
Description	Acronym	Description	Acronym
Bottom Surround Array Left	BSAL	---	---
Bottom Surround Array Right	BSAR	---	---
Bottom Back Left	BBL	---	---
Bottom Back Right	BBR	---	---
Bottom Back Center	BBC	---	---
Bottom Back Left Of Center	BBLC	---	---
Bottom Back Right Of Center	BBRC	---	---
Bottom Back Wide Left	BBWL	---	---
Bottom Back Wide Right	BBWR	---	---
Low Frequency Effects	LFE	Low Frequency Effects 1	LFE1
Low Frequency Effects Left	LFEL	---	---
Low Frequency Effects Right	LFER	Low Frequency Effects 2	LFE2
Headphone Left	HPL	---	---
Headphone Right	HPR	---	---

A.13 AMBISONIC COMPONENT ORDERING CONVENTION TYPES

Table A-13: Ambisonic Component Ordering Convention Types

Ambisonic Component Ordering Convention Type	Value
ORD_TYPE_UNDEFINED	0x00
AMBISONIC_CHANNEL_NUMBER (ACN)	0x01
FURSE_MALHAM	0x02
SINGLE_INDEX DESIGNATION (SID)	0x03

A.14 AMBISONIC NORMALIZATION TYPES

Table A-14: Ambisonic Normalization Types

Ambisonic Normalization Type	Value
NORM_TYPE_UNDEFINED	0x00
maxN	0x01
SN3D	0x02
N3D	0x03
SN2D	0x04
N2D	0x05

A.15 AUDIO CLASS-SPECIFIC AC INTERFACE DESCRIPTOR SUBTYPES

Table A-15: Audio Class-Specific AC Interface Descriptor Subtypes

Descriptor Subtype	Value
AC_DESCRIPTOR_UNDEFINED	0x00
HEADER	0x01
INPUT_TERMINAL	0x02
OUTPUT_TERMINAL	0x03
EXTENDED_TERMINAL	0x04
MIXER_UNIT	0x05
SELECTOR_UNIT	0x06
FEATURE_UNIT	0x07
EFFECT_UNIT	0x08
PROCESSING_UNIT	0x09
EXTENSION_UNIT	0x0A
CLOCK_SOURCE	0x0B
CLOCK_SELECTOR	0x0C
CLOCK_MULTIPLIER	0x0D
SAMPLE_RATE_CONVERTER	0x0E
CONNECTORS	0x0F
POWER_DOMAIN	0x10

A.16 AUDIO CLASS-SPECIFIC AS INTERFACE DESCRIPTOR SUBTYPES

Table A-16: Audio Class-Specific AS Interface Descriptor Subtypes

Descriptor Subtype	Value
AS_DESCRIPTOR_UNDEFINED	0x00
AS_GENERAL	0x01
AS_VALID_FREQ_RANGE	0x02

A.17 AUDIO CLASS-SPECIFIC STRING DESCRIPTOR SUBTYPES

Table A-17: Audio Class-Specific String descriptor Subtypes

Descriptor Subtype	Value
SUBTYPE_UNDEFINED	0x00

A.18 EXTENDED TERMINAL SEGMENT TYPES

Table A-18: Extended Terminal Segment Types

Segment Type	Value
SEGMENT_UNDEFINED	0x00
TERMINAL_VENDOR_DEFINED	0x1F
CHANNEL_BANDWIDTH	0x20
CHANNEL_MAGNITUDE_RESPONSE	0x21
CHANNEL_MAGNITUDE/PHASE_RESPONSE	0x22
CHANNEL_POSITION_XYZ	0x23
CHANNEL_POSITION_RØD	0x24
CHANNEL_VENDOR_DEFINED	0xFE
END_SEGMENT	0xFF

A.19 EFFECT UNIT EFFECT TYPES

Table A-19: Effect Unit Effect Types

wEffectType	Value
EFFECT_UNDEFINED	0x0000
PARAM_EQ_SECTION_EFFECT	0x0001
REVERBERATION_EFFECT	0x0002
MOD_DELAY_EFFECT	0x0003
DYN_RANGE_COMP_EFFECT	0x0004

A.20 PROCESSING UNIT PROCESS TYPES

Table A-20: Processing Unit Process Types

wProcessType	Value
PROCESS_UNDEFINED	0x0000
UP/DOWNMIX_PROCESS	0x0001
STEREO_EXTENDER_PROCESS	0x0002
MULTI_FUNCTION_PROCESS	0x0003

A.21 AUDIO CLASS-SPECIFIC ENDPOINT DESCRIPTOR SUBTYPES

Table A-21: Audio Class-Specific Endpoint Descriptor Subtypes

Descriptor Subtype	Value
DESCRIPTOR_UNDEFINED	0x00
EP_GENERAL	0x01

A.22 AUDIO CLASS-SPECIFIC REQUEST CODES

Table A-22: Audio Class-Specific Request Codes

Class-Specific Request Code	Value
REQUEST_CODE_UNDEFINED	0x00
CUR	0x01
RANGE	0x02
MEM	0x03
INTEN	0x04
STRING	0x05
HIGH_CAPABILITY_DESCRIPTOR	0x06

A.23 CONTROL SELECTOR CODES

A.23.1 AUDIOCONTROL INTERFACE CONTROL SELECTORS

Table A-23: AudioControl Interface Control Selectors

Control Selector	Value
AC_CONTROL_UNDEFINED	0x00
AC_ACTIVE_INTERFACE_CONTROL	0x01
AC_POWER_DOMAIN_CONTROL	0x02

A.23.2 CLOCK SOURCE CONTROL SELECTORS

Table A-24: Clock Source Control Selectors

Control Selector	Value
CS_CONTROL_UNDEFINED	0x00
CS_SAM_FREQ_CONTROL	0x01
CS_CLOCK_VALID_CONTROL	0x02

A.23.3 CLOCK SELECTOR CONTROL SELECTORS

Table A-25: Clock Selector Control Selectors

Control Selector	Value
CX_CONTROL_UNDEFINED	0x00
CX_CLOCK_SELECTOR_CONTROL	0x01

A.23.4 CLOCK MULTIPLIER CONTROL SELECTORS**Table A-26: Clock Multiplier Control Selectors**

Control Selector	Value
CM_CONTROL_UNDEFINED	0x00
CM_NUMERATOR_CONTROL	0x01
CM_DENOMINATOR_CONTROL	0x02

A.23.5 TERMINAL CONTROL SELECTORS**Table A-27: Terminal Control Selectors**

Control Selector	Value
TE_CONTROL_UNDEFINED	0x00
TE_INSERTION_CONTROL	0x01
TE_OVERLOAD_CONTROL	0x02
TE_UNDERFLOW_CONTROL	0x03
TE_OVERFLOW_CONTROL	0x04
TE_LATENCY_CONTROL	0x05

A.23.6 MIXER CONTROL SELECTORS**Table A-28: Mixer Control Selectors**

Control Selector	Value
MU_CONTROL_UNDEFINED	0x00
MU_MIXER_CONTROL	0x01
MU_UNDERFLOW_CONTROL	0x02
MU_OVERFLOW_CONTROL	0x03
MU_LATENCY_CONTROL	0x04

A.23.7 SELECTOR CONTROL SELECTORS**Table A-29: Selector Control Selectors**

Control Selector	Value
SU_CONTROL_UNDEFINED	0x00
SU_SELECTOR_CONTROL	0x01
SU_LATENCY_CONTROL	0x02

A.23.8 FEATURE UNIT CONTROL SELECTORS

Table A-30: Feature Unit Control Selectors

Control Selector	Value
FU_CONTROL_UNDEFINED	0x00
FU_MUTE_CONTROL	0x01
FU_VOLUME_CONTROL	0x02
FU_BASS_CONTROL	0x03
FU_MID_CONTROL	0x04
FU_TREBLE_CONTROL	0x05
FU_GRAPHIC_EQUALIZER_CONTROL	0x06
FU_AUTOMATIC_GAIN_CONTROL	0x07
FU_DELAY_CONTROL	0x08
FU_BASS_BOOST_CONTROL	0x09
FU_LOUDNESS_CONTROL	0x0A
FU_INPUT_GAIN_CONTROL	0x0B
FU_INPUT_GAIN_PAD_CONTROL	0x0C
FU_PHASE_INVERTER_CONTROL	0x0D
FU_UNDERFLOW_CONTROL	0x0E
FU_OVERFLOW_CONTROL	0x0F
FU_LATENCY_CONTROL	0x10

A.23.9 EFFECT UNIT CONTROL SELECTORS

A.23.9.1 PARAMETRIC EQUALIZER SECTION EFFECT UNIT CONTROL SELECTORS

Table A-31: Reverberation Effect Unit Control Selectors

Control Selector	Value
PE_CONTROL_UNDEFINED	0x00
PE_ENABLE_CONTROL	0x01
PE_CENTERFREQ_CONTROL	0x02
PE_QFACTOR_CONTROL	0x03
PE_GAIN_CONTROL	0x04
PE_UNDERFLOW_CONTROL	0x05
PE_OVERFLOW_CONTROL	0x06
PE_LATENCY_CONTROL	0x07

A.23.9.2 REVERBERATION EFFECT UNIT CONTROL SELECTORS

Table A-32: Reverberation Effect Unit Control Selectors

Control Selector	Value
RV_CONTROL_UNDEFINED	0x00
RV_ENABLE_CONTROL	0x01
RV_TYPE_CONTROL	0x02
RV_LEVEL_CONTROL	0x03
RV_TIME_CONTROL	0x04
RV_FEEDBACK_CONTROL	0x05
RV_PREDELAY_CONTROL	0x06
RV_DENSITY_CONTROL	0x07
RV_HIFREQ_ROLLOFF_CONTROL	0x08
RV_UNDERFLOW_CONTROL	0x09
RV_OVERFLOW_CONTROL	0x0A
RV_LATENCY_CONTROL	0x0B

A.23.9.3 MODULATION DELAY EFFECT UNIT CONTROL SELECTORS

Table A-33: Modulation Delay Effect Unit Control Selectors

Control Selector	Value
MD_CONTROL_UNDEFINED	0x00
MD_ENABLE_CONTROL	0x01
MD_BALANCE_CONTROL	0x02
MD_RATE_CONTROL	0x03
MD_DEPTH_CONTROL	0x04
MD_TIME_CONTROL	0x05
MD_FEEDBACK_CONTROL	0x06
MD_UNDERFLOW_CONTROL	0x07
MD_OVERFLOW_CONTROL	0x08
MD_LATENCY_CONTROL	0x09

A.23.9.4 DYNAMIC RANGE COMPRESSOR EFFECT UNIT CONTROL SELECTORS

Table A-34: Dynamic Range Compressor Effect Unit Control Selectors

Control Selector	Value
DR_CONTROL_UNDEFINED	0x00
DR_ENABLE_CONTROL	0x01
DR_COMPRESSION_RATE_CONTROL	0x02
DR_MAXAMPL_CONTROL	0x03
DR_THRESHOLD_CONTROL	0x04
DR_ATTACK_TIME_CONTROL	0x05
DR_RELEASE_TIME_CONTROL	0x06

Control Selector	Value
DR_UNDERFLOW_CONTROL	0x07
DR_OVERFLOW_CONTROL	0x08
DR_LATENCY_CONTROL	0x09

A.23.10 PROCESSING UNIT CONTROL SELECTORS

A.23.10.1 UP/DOWN-MIX PROCESSING UNIT CONTROL SELECTORS

Table A-35: Up/Down-mix Processing Unit Control Selectors

Control Selector	Value
UD_CONTROL_UNDEFINED	0x00
UD_MODE_SELECT_CONTROL	0x01
UD_UNDERFLOW_CONTROL	0x02
UD_OVERFLOW_CONTROL	0x03
UD_LATENCY_CONTROL	0x04

A.23.10.2 STEREO EXTENDER PROCESSING UNIT CONTROL SELECTORS

Table A-36: Stereo Extender Processing Unit Control Selectors

Control Selector	Value
ST_EXT_CONTROL_UNDEFINED	0x00
ST_EXT_WIDTH_CONTROL	0x01
ST_EXT_UNDERFLOW_CONTROL	0x02
ST_EXT_OVERFLOW_CONTROL	0x03
ST_EXT_LATENCY_CONTROL	0x04

A.23.11 EXTENSION UNIT CONTROL SELECTORS

Table A-37: Extension Unit Control Selectors

Control Selector	Value
XU_CONTROL_UNDEFINED	0x00
XU_UNDERFLOW_CONTROL	0x01
XU_OVERFLOW_CONTROL	0x02
XU_LATENCY_CONTROL	0x03

A.23.12 AUDIOSTREAMING INTERFACE CONTROL SELECTORS

Table A-38: AudioStreaming Interface Control Selectors

Control Selector	Value
AS_CONTROL_UNDEFINED	0x00
AS_ACT_ALT_SETTING_CONTROL	0x01
AS_VAL_ALT_SETTINGS_CONTROL	0x02
AS_AUDIO_DATA_FORMAT_CONTROL	0x03

A.23.13 ENDPOINT CONTROL SELECTORS

Table A-39: Endpoint Control Selectors

Control Selector	Value
EP_CONTROL_UNDEFINED	0x00
EP_PITCH_CONTROL	0x01
EP_DATA_OVERRUN_CONTROL	0x02
EP_DATA_UNDERRUN_CONTROL	0x03

A.24 CONNECTOR TYPES

Table A-40: Connector Types

Connector Type	Value
UNDEFINED	0x00
2.5 MM PHONE CONNECTOR	0x01
3.5 MM PHONE CONNECTOR	0x02
6.35 MM PHONE CONNECTOR	0x03
XLR/6.35MM COMBO CONNECTOR	0x04
XLR	0x05
OPTICAL/3.5MM COMBO CONNECTOR	0x06
RCA	0x07
BNC	0x08
BANANA	0x09
BINDING POST	0x0A
SPEAKON	0x0B
SPRING CLIP	0x0C
SCREW TYPE	0x0D
DIN	0x0E
MINI DIN	0x0F
EUROBLOCK	0x10
USB TYPE-C	0x11
RJ-11	0x12
RJ-45	0x13
TOSLINK	0x14
HDMI	0x15
Mini-HDMI	0x16
Micro-HDMI	0x17
DP	0x18
MINI-DP	0x19
D-SUB	0x1A
THUNDERBOLT	0x1B

Connector Type	Value
LIGHTNING	0x1C
WIRELESS	0x1D
USB STANDARD A	0x1E
USB STANDARD B	0x1F
USB MINI-B	0x20
USB MICRO-B	0x21
USB MICRO-AB	0x22
USB 3.0 MICRO-B	0x23
OTHER (CONNECTOR TYPE NOT IN THIS LIST)	0xFF

IECNORM.COM : Click to view the full PDF of IEC 62680-1-5:2019

IECNORM.COM : Click to view the full PDF of IEC 62680-1-5:2019

COMMISSION ÉLECTROTECHNIQUE INTERNATIONALE

INTERFACES DE BUS UNIVERSEL EN SÉRIE POUR LES DONNÉES ET L'ALIMENTATION ÉLECTRIQUE –

Partie 1-5: Composants communs – Définition de classes de dispositifs USB Audio 3.0

AVANT-PROPOS

- 1) La Commission Electrotechnique Internationale (IEC) est une organisation mondiale de normalisation composée de l'ensemble des comités électrotechniques nationaux (Comités nationaux de l'IEC). L'IEC a pour objet de favoriser la coopération internationale pour toutes les questions de normalisation dans les domaines de l'électricité et de l'électronique. A cet effet, l'IEC – entre autres activités – publie des Normes internationales, des Spécifications techniques, des Rapports techniques, des Spécifications accessibles au public (PAS) et des Guides (ci-après dénommés "Publication(s) de l'IEC"). Leur élaboration est confiée à des comités d'études, aux travaux desquels tout Comité national intéressé par le sujet traité peut participer. Les organisations internationales, gouvernementales et non gouvernementales, en liaison avec l'IEC, participent également aux travaux. L'IEC collabore étroitement avec l'Organisation Internationale de Normalisation (ISO), selon des conditions fixées par accord entre les deux organisations.
- 2) Les décisions ou accords officiels de l'IEC concernant les questions techniques représentent, dans la mesure du possible, un accord international sur les sujets étudiés, étant donné que les Comités nationaux de l'IEC intéressés sont représentés dans chaque comité d'études.
- 3) Les Publications de l'IEC se présentent sous la forme de recommandations internationales et sont agréées comme telles par les Comités nationaux de l'IEC. Tous les efforts raisonnables sont entrepris afin que l'IEC s'assure de l'exactitude du contenu technique de ses publications; l'IEC ne peut pas être tenue responsable de l'éventuelle mauvaise utilisation ou interprétation qui en est faite par un quelconque utilisateur final.
- 4) Dans le but d'encourager l'uniformité internationale, les Comités nationaux de l'IEC s'engagent, dans toute la mesure possible, à appliquer de façon transparente les Publications de l'IEC dans leurs publications nationales et régionales. Toutes divergences entre toutes Publications de l'IEC et toutes publications nationales ou régionales correspondantes doivent être indiquées en termes clairs dans ces dernières.
- 5) L'IEC elle-même ne fournit aucune attestation de conformité. Des organismes de certification indépendants fournissent des services d'évaluation de conformité et, dans certains secteurs, accèdent aux marques de conformité de l'IEC. L'IEC n'est responsable d'aucun des services effectués par les organismes de certification indépendants.
- 6) Tous les utilisateurs doivent s'assurer qu'ils sont en possession de la dernière édition de cette publication.
- 7) Aucune responsabilité ne doit être imputée à l'IEC, à ses administrateurs, employés, auxiliaires ou mandataires, y compris ses experts particuliers et les membres de ses comités d'études et des Comités nationaux de l'IEC, pour tout préjudice causé en cas de dommages corporels et matériels, ou de tout autre dommage de quelque nature que ce soit, directe ou indirecte, ou pour supporter les coûts (y compris les frais de justice) et les dépenses découlant de la publication ou de l'utilisation de cette Publication de l'IEC ou de toute autre Publication de l'IEC, ou au crédit qui lui est accordé.
- 8) L'attention est attirée sur les références normatives citées dans cette publication. L'utilisation de publications référencées est obligatoire pour une application correcte de la présente publication.
- 9) L'attention est attirée sur le fait que certains des éléments de la présente Publication de l'IEC peuvent faire l'objet de droits de brevet. L'IEC ne saurait être tenue pour responsable de ne pas avoir identifié de tels droits de brevets et de ne pas avoir signalé leur existence.

La Norme internationale IEC 62680-1-5 a été établie par le Domaine technique 18: Systèmes multimédias domestiques et applications pour réseaux d'utilisateurs finaux, du comité d'études 100 de l'IEC: Systèmes et équipements audio, vidéo et services de données.

Le texte de cette norme a été élaboré par l'USB Implementers Forum (USB-IF). Les règles structurelles et rédactionnelles utilisées dans la présente publication reflètent les pratiques en vigueur au sein de l'organisme responsable de sa soumission.

Le texte de cette Norme internationale est issu des documents suivants:

CDV	Rapport de vote
100/3157/CDV	100/3227/RVC

Le rapport de vote indiqué dans le tableau ci-dessus donne toute information sur le vote ayant abouti à l'approbation de cette Norme internationale.

Le comité a décidé que le contenu de ce document ne sera pas modifié avant la date de stabilité indiquée sur le site web de l'IEC sous "<http://webstore.iec.ch>" dans les données relatives au document recherché. A cette date, le document sera

- reconduit,
- supprimé,
- remplacé par une édition révisée, ou
- amendé.

IMPORTANT – Le logo "*colour inside*" qui se trouve sur la page de couverture de cette publication indique qu'elle contient des couleurs qui sont considérées comme utiles à une bonne compréhension de son contenu. Les utilisateurs devraient, par conséquent, imprimer cette publication en utilisant une imprimante couleur.

INTRODUCTION

La série IEC 62680 est issue d'une série de spécifications initialement établies par l'USB Implementers Forum (USB-IF). Ces spécifications ont été soumises à l'IEC dans le cadre d'un accord particulier conclu entre l'IEC et l'USB-IF.

La présente norme est la publication relative à la définition de classes de dispositifs USB pour dispositifs audio, version 3.0, publiée par l'USB-IF.

L'USB Implementers Forum, Inc. (USB-IF) est un organisme à but non lucratif fondé par le groupe de sociétés qui a développé la spécification du bus universel en série. L'USB-IF a été créé dans le but de proposer un organisme et un forum à même de favoriser la progression et l'adoption de la technologie USB. Le forum facilite le développement de périphériques (dispositifs) USB compatibles et de haute qualité et promeut les avantages de la technologie USB et la qualité des produits qui ont été validés par des essais de conformité.

TOUTES LES SPÉCIFICATIONS USB VOUS SONT FOURNIES "EN L'ÉTAT" SANS GARANTIE D'AUCUNE SORTE, Y COMPRIS TOUTE GARANTIE DE QUALITÉ MARCHANDE, DE NON-VIOLATION OU D'ADÉQUATION À UN USAGE PARTICULIER. L'USB IMPLEMENTERS FORUM ET LES AUTEURS DE L'ENSEMBLE DES SPÉCIFICATIONS USB DÉCLINENT TOUTE RESPONSABILITÉ, Y COMPRIS TOUTE RESPONSABILITÉ RELATIVE À LA VIOLATION DE DROITS DE PROPRIÉTÉ, EN CE QUI CONCERNE L'UTILISATION OU LA MISE EN ŒUVRE DES INFORMATIONS CONTENUES DANS LA PRÉSENTE SPÉCIFICATION.

LA MISE À DISPOSITION D'UNE SPÉCIFICATION USB, QUELLE QU'ELLE SOIT, N'IMPLIQUE L'OCTROI D'AUCUNE LICENCE, EXPRESSE OU IMPLICITE, PAR PERCLUSION OU AUTRE, SUR AUCUN DROIT DE PROPRIÉTÉ INTELLECTUELLE.

La conclusion des accords des adoptants de l'USB peut toutefois permettre à une société signataire de participer à un accord de licence réciproque RAND-Z pour les produits conformes. Pour plus d'informations, voir:

<https://www.usb.org/documents>

L'IEC NE PREND AUCUNE POSITION QUANT À SAVOIR S'IL VOUS EST CONSEILLÉ DE CONCLURE UN QUELCONQUE ACCORD DES ADOPTANTS DE L'USB OU DE PARTICIPER À L'USB IMPLEMENTERS FORUM.

DEFINITION DE CLASSES DE DISPOSITIFS DE BUS UNIVERSEL EN SERIE POUR DISPOSITIFS AUDIO

IECNORM.COM : Click to view the full PDF of IEC 62680-1-5:2019

Version 3.0

22 septembre 2016

DOMAINE D'APPLICATION DE LA PRESENTE VERSION

Le présent document est la Version 3.0 de la présente définition de classes de dispositifs.

COLLABORATEURS

Joe Scanlon	Advanced Micro Devices
Rhoads Hollowell	Apple Inc.
Girault Jones	Apple Inc.
Matthew X. Mora	Apple Inc.
Tzung-Dar Tsai	C-Media Electronics, Inc.
Brad Lambert	Cirrus Logic, Inc.
Dan Bogard	Conexant Systems, Inc.
Pete Burgers	DisplayLink (UK), Ltd.
David Roh	Dolby Laboratories, Inc.
Leng Ooi	Google, Inc.
Pierre-Louis Bossart	Intel Corporation
David Hines	Intel Corporation
Abdul Rahman Ismail (Co-président)	Intel Corporation
Devon Worrell	Intel Corporation
Chandrashekhar Rao	Logitech, Inc.
Terry Moore	MCCI Corporation
Alex Lin	MediaTek, Inc.
Bala Sivakumar	Microsoft Corporation
Geert Knapen (Co-président et éditeur)	NXP Semiconductors
	PL Mobile Audio
	411 E. Plumeria drive
	San Jose, CA 95134, USA
	E-mail: geert.knapen@nxp.com
James Goel	Qualcomm, Inc.
Andre Schevciw	Qualcomm, Inc.
Jin-Sheng Wang	Qualcomm, Inc.
Morten Christiansen	Synopsys

HISTORIQUE DES REVISIONS

Révision	Date	Nom de fichier	Description
1.0	18 mars 1998	Audio10.pdf	Version 1.0
2.0	31 mai 2006	Audio20 final.pdf	Version 2.0
3.0	22 septembre 2016	Audio30.pdf	Version 3.0

**Copyright © 1997-2016 USB Implementers Forum, Inc.
Tous droits réservés.**

DENI DE RESPONSABILITÉ CONCERNANT LA PROPRIÉTÉ INTELLECTUELLE

LE PRÉSENT DOCUMENT ACCORDE UNE LICENCE POUR LA REPRODUCTION DE LA PRÉSENTE SPÉCIFICATION POUR UN USAGE INTERNE UNIQUEMENT. LE PRÉSENT DOCUMENT N'ACCORDE NI NE VISE À ACCORDER AUCUNE AUTRE LICENCE, EXPRESSE OU IMPLICITE, PAR PERCLUSION OU AUTRE.

L'USB-IF ET LES AUTEURS DE LA PRÉSENTE SPÉCIFICATION DÉCLINENT EXPRESSÉMENT TOUTE RESPONSABILITÉ RELATIVE À LA VIOLATION DE DROITS DE PROPRIÉTÉ INTELLECTUELLE EN CE QUI CONCERNE LA MISE EN ŒUVRE DES INFORMATIONS CONTENUES DANS LA PRÉSENTE SPÉCIFICATION. EN OUTRE, L'USB-IF ET LES AUTEURS DE LA PRÉSENTE SPÉCIFICATION NE GARANTISSENT NI NE DÉCLARENT QUE LA OU LESDITES MISES EN ŒUVRE NE VIOLERONT PAS LES DROITS DE PROPRIÉTÉ INTELLECTUELLE DE TIERS.

LA PRÉSENTE SPÉCIFICATION EST FOURNIE "EN L'ÉTAT", SANS GARANTIE, EXPRESSE OU IMPLICITE, LÉGALE OU AUTRE. TOUTES LES GARANTIES SONT EXPRESSÉMENT EXCLUES. L'USB-IF, LES MEMBRES DE L'USB-IF ET LES AUTEURS DE LA PRÉSENTE SPÉCIFICATION N'OFFRENT AUCUNE GARANTIE DE QUALITÉ MARCHANDE, AUCUNE GARANTIE DE NON-VIOLATION, AUCUNE GARANTIE D'ADAPTATION À UN USAGE PARTICULIER NI AUCUNE GARANTIE ÉMANANT D'UNE PROPOSITION, D'UNE SPÉCIFICATION OU D'UN ÉCHANTILLON QUELCONQUES.

L'USB-IF, LES MEMBRES DE L'USB-IF ET LES AUTEURS NE POURRONT EN AUCUN CAS ÊTRE REDEVABLES À UN TIERS DU COÛT D'ACQUISITION DE BIENS OU DE SERVICES DE REMPLACEMENT, D'UN MANQUE À GAGNER, D'UNE PRIVATION DE JOUISSANCE, D'UNE PERTE DE DONNÉES OU DE TOUT DOMMAGE ACCESSOIRE, CONSÉCUTIF, INDIRECT OU PARTICULIER, EN VERTU D'UN CONTRAT, D'UN DÉLIT, D'UNE GARANTIE OU AUTRE, ÉMANANT DE QUELQUE MANIÈRE QUE CE SOIT DE L'UTILISATION DE LA PRÉSENTE SPÉCIFICATION, QUE LEDIT TIERS AIT OU NON ÉTÉ AVISÉ AU PRÉALABLE DE LA POSSIBILITÉ DE TELS DOMMAGES.

NOTE: PLUSIEURS MEMBRES DE L'USB-IF ONT PARTICIPÉ À L'ÉLABORATION DE LA PRÉSENTE SPÉCIFICATION. CERTAINS DE CES MEMBRES PEUVENT AVOIR REFUSÉ DE CONCLURE UN ACCORD SPÉCIFIQUE QUANT À L'OCTROI D'UNE LICENCE CONCERNANT LES DROITS DE PROPRIÉTÉ INTELLECTUELLE POUVANT FAIRE L'OBJET D'UNE VIOLATION LORS DE LA MISE EN ŒUVRE DE LA PRÉSENTE SPÉCIFICATION. LES PERSONNES METTANT EN ŒUVRE LA PRÉSENTE SPÉCIFICATION EN ASSUMENT LES RISQUES.

Dolby™, AC-3™, Pro Logic™ et Dolby Surround™ sont des marques déposées de Dolby Laboratories, Inc.

Tous les autres noms de produits sont des marques, des marques déposées ou des marques de service de leurs propriétaires respectifs.

Envoyer les commentaires par courrier électronique à audio-chair@usb.org

SOMMAIRE

Domaine d'application de la présente version	158
Collaborateurs.....	158
Historique des révisions	158
Sommaire	160
Tableaux	164
Figures	166
1 Introduction	168
1.1 Domaine d'application	168
1.2 Objectif.....	168
1.3 Documents connexes	168
1.4 Termes et abréviations.....	168
2 Vue d'ensemble de gestion	172
2.1 Vue d'ensemble des principales différences entre ADC v2.0 et v3.0.....	172
3 Caractéristiques fonctionnelles.....	174
3.1 Introduction	174
3.2 Définition de dispositif audio de base.....	177
3.3 Rétrocompatibilité	177
3.4 Association d'interfaces audio (AIA) et descripteur d'association d'interfaces	178
3.4.1 Classe de fonction audio	178
3.4.2 Sous-classe de fonction audio	178
3.4.3 Protocole de fonction audio.....	178
3.5 Classe d'interface audio	178
3.6 Sous-classe d'interface audio.....	179
3.7 Protocole d'interface audio.....	179
3.8 Catégorie de fonction audio.....	179
3.9 Domaines d'horloge	180
3.10 Domaines d'alimentation	180
3.11 Types de synchronisation audio.....	180
3.11.1 Asynchrone.....	181
3.11.2 Synchrone.....	181
3.11.3 Adaptative	181
3.11.4 Implications des différents types de synchronisation.....	181
3.12 Synchronisation entre les canaux	183
3.13 Topologie de fonction audio	183
3.13.1 Grappe.....	188
3.13.2 Terminal d'entrée.....	189
3.13.3 Terminal de sortie	190
3.13.4 Unité de mixage	190
3.13.5 Unité de sélection	191

3.13.6	Unité de fonction	191
3.13.7	Unité de conversion du débit d'échantillonnage	192
3.13.8	Unité d'effet	193
3.13.9	Unité de traitement	197
3.13.10	Unité d'extension	199
3.13.11	Entités d'horloge	200
3.14	Modèle opérationnel	202
3.14.1	Interface AudioControl.....	203
3.14.2	Interface AudioStreaming	203
3.14.3	Modèle d'horloge.....	205
3.14.4	Modèle de domaines d'alimentation	206
3.14.5	Considérations et exigences supplémentaires relatives à l'alimentation	207
3.14.6	Liaison entre les boutons physiques et les commandes audio	208
4	Descripteurs	209
4.1	Descripteurs normalisés.....	209
4.2	Descripteurs spécifiques à la classe	209
4.2.1	Descripteurs traditionnels spécifiques à la classe	210
4.2.2	Descripteurs haute capacité spécifiques à la classe.....	210
4.3	Descripteur de grappe.....	211
4.3.1	En-tête de descripteur de grappe	212
4.3.2	Bloc de descripteur de grappe	213
4.3.3	Exemple de descripteur de grappe.....	221
4.3.4	CEA-861.2 Mise en correspondance des canaux.....	222
4.4	Grappe physique et grappe logique.....	222
4.4.1	Mise en correspondance entre grappes physiques et logiques.....	223
4.5	Descripteurs de l'interface AudioControl.....	226
4.5.1	Descripteur d'interface AC normalisé	226
4.5.2	Descripteur d'interface AC spécifique à la classe.....	226
4.6	Descripteurs de point d'extrémité AudioControl.....	260
4.6.1	Descripteurs de points d'extrémité de commande AC	260
4.6.2	Descripteurs de point d'extrémité d'interruption AC	260
4.7	Descripteurs de l'interface AudioStreaming	261
4.7.1	Descripteur d'interface AS normalisé.....	261
4.7.2	Descripteur d'interface AS spécifique à la classe	262
4.7.3	Descripteur de plage de fréquences valide AS spécifique à la classe	263
4.8	Descripteurs de point d'extrémité AudioStreaming	264
4.8.1	Descripteurs de point d'extrémité de données audio isochrone AS.....	264
4.8.2	Descripteur de point d'extrémité de rétroaction isochrone AS	266
4.9	Descripteurs de chaîne spécifiques à la classe.....	267
5	Demandes	269
5.1	Demandes normalisées.....	269

5.2	Demandes spécifiques à la classe	269
5.2.1	Demandes AudioControl	270
5.2.2	Demandes AudioStreaming.....	298
5.2.3	Demandes supplémentaires.....	301
6	Interruptions	305
6.1	Message de données d'interruption	305
6.2	Sources d'interruption	307
Annexe A.	Codes de classes de dispositifs audio.....	308
A.1	Code de classe de fonction audio.....	308
A.2	Codes de sous-classes de fonction audio	308
A.3	Codes de protocoles de fonction audio	308
A.4	Code de classe d'interface audio.....	308
A.5	Codes de sous-classes d'interface audio	309
A.6	Codes de protocoles d'interface audio	309
A.7	Codes de catégories de fonction audio.....	309
A.8	Types de descripteurs spécifiques à une classe audio	310
A.9	Sous-types de descripteurs de grappes	310
A.10	Types de segments de descripteurs de grappes	310
A.11	Définitions des objectifs des canaux	311
A.12	Définitions des relations de canaux	311
A.13	Types de conventions d'ordre des composants d'ambiophonie.....	313
A.14	Types de normalisations d'ambiophonie.....	314
A.15	Sous-types de descripteurs d'interface AC spécifiques à la classe audio.....	314
A.16	Sous-types de descripteurs d'interface AS spécifiques à la classe audio	315
A.17	Sous-types de descripteurs de chaîne spécifiques à la classe audio.....	315
A.18	Types de segments de terminal étendus	315
A.19	Types d'effets d'unité d'effet	315
A.20	Types de processus d'unité de traitement	316
A.21	Sous-types de descripteurs de points d'extrémité spécifiques à la classe audio.....	316
A.22	Codes de demandes spécifiques à la classe audio	316
A.23	Codes de sélecteurs de commande	316
A.23.1	Sélecteurs de commande d'interface AudioControl	316
A.23.2	Sélecteurs de commande de source d'horloge	317
A.23.3	Sélecteurs de commande de sélecteur d'horloge	317
A.23.4	Sélecteurs de commande de multiplicateur d'horloge	317
A.23.5	Sélecteurs de commande de terminal	317
A.23.6	Sélecteurs de commande de mixage	317
A.23.7	Sélecteurs de commande de sélection.....	318
A.23.8	Sélecteurs de commande d'unité de fonction	318
A.23.9	Sélecteurs de commande d'unité d'effet	318
A.23.10	Sélecteurs de commande d'unité de traitement	320

A.23.11	Sélecteurs de commande d'unité d'extension	321
A.23.12	Sélecteurs de commande d'interface AudioStreaming.....	321
A.23.13	Sélecteurs de commande de point d'extrémité.....	321
A.24	Types de connecteurs	321

IECNORM.COM : Click to view the full PDF of IEC 62680-1-5:2019

TABLEAUX

Tableau 4-1: Topologie des descripteurs traditionnels spécifiques à la classe.....	210
Tableau 4-2: Topologie des descripteurs haute capacité spécifiques à la classe.....	211
Tableau 4-3: En-tête de descripteur de grappe	213
Tableau 4-4: Segment de descripteur de grappe.....	214
Tableau 4-5: Segment de fin	214
Tableau 4-6: Segment de description de grappe	214
Tableau 4-7: Segment défini par le fournisseur.....	215
Tableau 4-8: Relations des canaux.....	216
Tableau 4-9: Segment d'informations.....	219
Tableau 4-10: Segment d'ambiphonie.....	220
Tableau 4-11: Segment de description de canal	220
Tableau 4-12: Segment défini par le fournisseur	220
Tableau 4-13: Exemple de descripteur de grappe	221
Tableau 4-14: Descripteur d'interface AC normalisé	226
Tableau 4-15: Descripteur d'en-tête d'interface AC spécifique à la classe.....	227
Tableau 4-16: Descripteur de terminal d'entrée.....	229
Tableau 4-17: Descripteur de terminal de sortie	231
Tableau 4-18: En-tête du descripteur de terminal étendu	233
Tableau 4-19: Segment de descripteur de grappe.....	234
Tableau 4-20: Segment de fin	234
Tableau 4-21: Segment défini par le fournisseur.....	234
Tableau 4-22: Segment de bande passante.....	235
Tableau 4-23: Segment d'amplitude.....	235
Tableau 4-24: Segment d'amplitude/phase.....	236
Tableau 4-25: Segment de position	236
Tableau 4-26: Segment de position	237
Tableau 4-27: Segment défini par le fournisseur.....	237
Tableau 4-28: Descripteur de connecteurs.....	239
Tableau 4-29: Descripteur d'unité de mixage	242
Tableau 4-30: Descripteur d'unité de sélection	243
Tableau 4-31: Descripteur d'unité de fonction	244
Tableau 4-32: Descripteur d'unité de conversion du débit d'échantillonnage	245
Tableau 4-33: Descripteur d'unité d'effet.....	246
Tableau 4-34: Descripteur d'unité d'effet de section d'égaliseur paramétrique.....	247
Tableau 4-35: Descripteur d'unité d'effet de réverbération.....	248
Tableau 4-36: Descripteur d'unité d'effet de retard de modulation	249
Tableau 4-37: Descripteur d'unité d'effet de compresseur de plage dynamique.....	250
Tableau 4-38: Partie commune du descripteur d'unité de traitement.....	252
Tableau 4-39: Descripteur d'unité de traitement de mixage élévateur/réducteur	253

Tableau 4-40: Descripteur d'unité de traitement d'amplification stéréo	254
Tableau 4-41: Descripteur d'unité de traitement multifonction	255
Tableau 4-42: Descripteur d'unité d'extension	256
Tableau 4-43: Descripteur de source d'horloge	257
Tableau 4-44: Descripteur de sélecteur d'horloge	258
Tableau 4-45: Descripteur de multiplicateur d'horloge	259
Tableau 4-46: Descripteur de domaine d'alimentation	260
Tableau 4-47: Descripteur de point d'extrémité d'interruption AC normalisé	261
Tableau 4-48: Descripteur d'interface AS normalisé	262
Tableau 4-49: Descripteur d'interface AS spécifique à la classe	263
Tableau 4-50: Descripteur de plage de fréquences valide AS spécifique à la classe	264
Tableau 4-51: Descripteur de point d'extrémité de données audio isochrone AS normalisé	265
Tableau 4-52: Descripteur de point d'extrémité de données audio isochrone AS spécifique à la classe	266
Tableau 4-53: Descripteur de point d'extrémité de rétroaction isochrone AS normalisé	267
Tableau 4-54: Descripteur de chaîne spécifique à la classe	268
Tableau 5-1: Topologie de demande	271
Tableau 5-2: Bloc de paramètre CUR d'une commande de 1 octet	273
Tableau 5-3: Bloc de paramètre RANGE d'une commande de 1 octet	273
Tableau 5-4: Bloc de paramètre CUR d'une commande de 2 octets	274
Tableau 5-5: Bloc de paramètre RANGE d'une commande de 2 octets	274
Tableau 5-6: Bloc de paramètre CUR d'une commande de 4 octets	274
Tableau 5-7: Bloc de paramètre RANGE d'une commande de 4 octets	275
Tableau 5-8: Bloc de paramètre INTEN	275
Tableau 5-9: Bloc de paramètre CUR d'une commande d'insertion	278
Tableau 5-10: Numéros de bande et fréquences médianes (norme ANSI S1.11-1986)	283
Tableau 5-11: Bloc de paramètre CUR d'une commande d'égaliseur graphique	284
Tableau 5-12: Bloc de paramètre RANGE d'une commande d'égaliseur graphique	285
Tableau 5-13: Bloc de paramètre CUR d'une commande de paramètre alternatif valide	299
Tableau 5-14: Bloc de paramètre CUR d'une commande de format de données audio	300
Tableau 5-15: Valeurs de demande de mémoire	301
Tableau 5-16: Demande de chaîne	302
Tableau 5-17: Demande de descripteur haute capacité	303
Tableau 6-1: Format du message de données d'interruption	307
Tableau A-1: Code de classe de fonction audio	308
Tableau A-2: Codes de sous-classes de fonction audio	308
Tableau A-3: Codes de protocoles de fonction audio	308
Tableau A-4: Code de classe d'interface audio	308
Tableau A-5: Codes de sous-classes d'interface audio	309
Tableau A-6: Codes de protocoles d'interface audio	309
Tableau A-7: Codes de catégories de fonction audio	309
Tableau A-8: Types de descripteurs spécifiques à la classe audio	310

Tableau A-9: Sous-types de descripteurs de grappes spécifiques à la classe audio	310
Tableau A-10: Types de segments de descripteurs de grappes	310
Tableau A-11: Définitions des objectifs des canaux.....	311
Tableau A-12: Définitions des relations de canaux.....	311
Tableau A-13: Types de conventions d'ordre des composants d'ambiophonie	313
Tableau A-14: Types de normalisations d'ambiophonie.....	314
Tableau A-15: Sous-types de descripteurs d'interface AC spécifiques à la classe audio	314
Tableau A-16: Sous-types de descripteurs d'interface AS spécifiques à la classe audio.....	315
Tableau A-17: Sous-types de descripteurs de chaîne spécifiques à la classe audio	315
Tableau A-18: Types de segments de terminal étendus	315
Tableau A-19: Types d'effets d'unité d'effet	315
Tableau A-20: Types de processus d'unité de traitement.....	316
Tableau A-21: Sous-types de descripteurs de points d'extrémité spécifiques à la classe audio	316
Tableau A-22: Codes de demandes spécifiques à la classe audio	316
Tableau A-23: Sélecteurs de commande d'interface AudioControl.....	316
Tableau A-24: Sélecteurs de commande de source d'horloge.....	317
Tableau A-25: Sélecteurs de commande de sélecteur d'horloge.....	317
Tableau A-26: Sélecteurs de commande de multiplicateur d'horloge.....	317
Tableau A-27: Sélecteurs de commande de terminal	317
Tableau A-28: Sélecteurs de commande de mixage	317
Tableau A-29: Sélecteurs de commande de sélection	318
Tableau A-30: Sélecteurs de commande d'unité de fonction	318
Tableau A-31: Sélecteurs de commande d'unité d'effet de réverbération.....	318
Tableau A-32: Sélecteurs de commande d'unité d'effet de réverbération.....	319
Tableau A-33: Sélecteurs de commande d'unité d'effet de retard de modulation	319
Tableau A-34: Sélecteurs de commande d'unité d'effet de compresseur de plage dynamique	320
Tableau A-35: Sélecteurs de commande d'unité de traitement de mixage élévateur/réducteur.....	320
Tableau A-36: Sélecteurs de commande d'unité de traitement d'amplification stéréo	320
Tableau A-37: Sélecteurs de commande d'unité d'extension.....	321
Tableau A-38: Sélecteurs de commande d'interface AudioStreaming	321
Tableau A-39: Sélecteurs de commande de point d'extrémité.....	321
Tableau A-40: Types de connecteurs	321

FIGURES

Figure 3-1: Plusieurs configurations et leurs interfaces et associations d'interfaces audio	174
Figure 3-2: Vue d'ensemble de la fonction audio	176
Figure 3-3: Au sein de la fonction audio	187
Figure 3-4: Icône du terminal d'entrée	189
Figure 3-5: Icône du terminal de sortie.....	190
Figure 3-6: Icône de l'unité de mixage	191

Figure 3-7: Icône de l'unité de sélection	191
Figure 3-8: Icône de l'unité de fonction	192
Figure 3-9: Icône de l'unité de conversion du débit d'échantillonnage	193
Figure 3-10: Icône de l'unité d'effet PEQS	194
Figure 3-11: Icône de l'unité d'effet de réverbération.....	195
Figure 3-12: Icône de l'unité d'effet de retard de modulation	195
Figure 3-13: Caractéristique de transfert du compresseur de plage dynamique	196
Figure 3-14: Icône de l'unité d'effet de compresseur de plage dynamique	197
Figure 3-15: Icône de l'unité de traitement de mixage élévateur/réducteur	198
Figure 3-16: Icône de l'unité de traitement d'amplification stéréo	198
Figure 3-17 Icône de l'unité de traitement multifonction	199
Figure 3-18: Icône de l'unité d'extension.....	200
Figure 3-19: Icône de la source d'horloge.....	201
Figure 3-20: Icône du sélecteur d'horloge	201
Figure 3-21: Icône du multiplicateur d'horloge.....	202
Figure 4-1: Descripteur de grappe	212
Figure 4-2: Bloc de descripteur de grappe	213
Figure 4-3: Représentation en 3D des relations des canaux.....	218
Figure 4-4: Mise en correspondance d'une grappe physique vers une grappe logique	224
Figure 4-5: Mise en correspondance d'une grappe logique vers une grappe physique	225
Figure 4-6: Descripteur de terminal étendu	232
Figure 4-7: Bloc de canal de terminal étendu	233
Figure 4-8: Composantes internes de l'unité de mixage.....	241

1 INTRODUCTION

1.1 DOMAINE D'APPLICATION

La définition de classes de dispositifs audio s'applique à tous les dispositifs ou fonctions intégrés dans des dispositifs composites qui sont utilisés pour manipuler des fonctionnalités audio, vocales et sonores. Cela inclut à la fois les données audio (analogiques et numériques) et les fonctionnalités utilisées pour contrôler directement l'environnement audio, comme les commandes de volume et de tonalité. La classe des dispositifs audio n'inclut pas de fonctionnalité permettant d'utiliser des mécanismes de transport liés à la reproduction de données audio, comme les mécanismes de transport de bande ou la commande d'un lecteur de CD-ROM.

1.2 OBJECTIF

Le présent document a pour objectif de décrire les capacités et caractéristiques minimales qu'un dispositif audio doit prendre en charge pour être conforme à l'USB. Le présent document fournit également des recommandations concernant les fonctionnalités facultatives.

1.3 DOCUMENTS CONNEXES

- *Spécification du bus universel en série*, révision 2.0 (appelée *Spécification USB* dans le présent document). Voir en particulier le Chapitre 5, "Modèle de flux de données USB" et le Chapitre 9, "Cadre de dispositif USB".
- *Universal Serial Bus 3.1 Specification*, révision 1.0 (disponible en anglais seulement) (appelée *Spécification USB 3.1* dans le présent document). Ce document couvre les détails spécifiques aux dispositifs SuperSpeed et SuperSpeed+.
- *Universal Serial Bus Device Class Definition for Audio Data Formats* (disponible en anglais seulement) (appelée *Formats de données audio USB* dans le présent document).
- *Universal Serial Bus Device Class Definition for Terminal Types* (disponible en anglais seulement) (appelée *Types de terminaux audio USB* dans le présent document).
- Norme ANSI S1.11-1986.
- Norme MPEG-1 ISO/IEC 111172-3 1993.
- Norme MPEG-2 ISO/IEC 13818-3, 20 février 1997.
- Norme de compression audionumérique (AC-3), ATSC A/52A, 20 août 2001 (disponible sur <http://www.atsc.org/>).
- ANSI/IEEE-754, norme de virgule flottante.
- Norme internationale ISO/IEC 60958: *Interface audionumérique et annexes*.
- Norme ISO/IEC 61937.

1.4 TERMES ET ABREVIATIONS

La présente section définit les termes utilisés dans le présent document. Pour les termes supplémentaires concernant le Bus universel en série, voir le Chapitre 2, "Termes et abréviations", de la *Spécification USB*.

ASRC

Acronyme d'Asynchronous Sampling Rate Conversion (conversion du débit d'échantillonnage asynchrone)

Grappe (de canaux audio)

Groupe de canaux audio logiques transportant des informations audio synchrones étroitement liées. Un flux audio stéréo est un exemple type de grappe de deux canaux.

Attribut de commande audio	Paramètre d'une commande audio. Les attributs actuel, minimum, maximum et de résolution d'une commande de volume en sont des exemples.
Commande audio	Objet logique utilisé pour manipuler une propriété audio spécifique. La commande de volume, la commande d'inhibition, etc. en sont des exemples.
Flux de données audio	Moyen de transport pouvant transporter des informations audio.
AEC	Acronyme d'Audio Echo Cancellation (annulation d'écho audio)
Fonction audio	Partie indépendante d'un dispositif USB qui traite des fonctionnalités audio.
Association d'interfaces audio (AIA)	Regroupement d'une interface AudioControl unique, d'aucune ou de plusieurs interfaces AudioStreaming et d'aucune ou de plusieurs interfaces, constituant ensemble une interface complète pour une version particulière (niveau de conformité) de la fonction audio.
Interface AudioControl (ACI - AudioControl interface)	Interface USB utilisée pour accéder aux commandes audio au sein d'une fonction audio.
Interface AudioStreaming (ASI - AudioStreaming interface)	Interface USB utilisée pour transporter des flux audio vers ou hors de la fonction audio.
Multiplicateur d'horloge (CM - Clock Multiplier)	Multiplie un signal d'horloge par un multiplicateur fractionnaire.
Sélecteur d'horloge (CX - Clock Selector)	Effectue un choix entre un certain nombre de signaux d'horloge en entrée.
Source d'horloge (CS - Clock Source)	Génère un signal d'horloge audio (horloge maître ou horloge d'échantillonnage).
CMD	Acronyme de Clock Multiplier descriptor (descripteur de multiplicateur d'horloge)
CSD	Acronyme de Clock Source descriptor (descripteur de source d'horloge)
CXD	Acronyme de Clock Selector descriptor (descripteur de sélecteur d'horloge)
Unité d'effet (EU - Effect Unit)	Permet une manipulation audio avancée sur les canaux audio logiques entrants.
Entité	Objet logique adressable au sein d'une fonction audio.
Unité d'extension (XU - Extension Unit)	Applique un processus non défini à un certain nombre de canaux d'entrée logiques.
Unité de fonction (FU - Feature Unit)	Permet une manipulation audio de base sur les canaux audio logiques entrants.

FUD	Acronyme de Feature Unit descriptor (descripteur d'unité de fonction)
Descripteur haute capacité	Nouvelle représentation pour les descripteurs spécifiques à la classe permettant des descripteurs potentiellement importants (>256 octets) et dynamiques.
Broche d'entrée	Connexion d'entrée logique à une entité. Transporte une grappe unique.
Terminal d'entrée (IT - Input Terminal)	Réceptacle pour les informations audio entrant dans la fonction audio.
ITD	Acronyme d'Input Terminal descriptor (descripteur de terminal d'entrée)
Canal audio logique	Moyen de transport logique pour un canal audio unique. Fait abstraction des propriétés physiques et des formats de la connexion. Est généralement identifié par localisation spatiale. Par exemple, canal gauche, canal surround droite, etc.
Interface MIDISstreaming (MSI - MIDISstreaming interface)	Interface USB qui peut être utilisée pour transporter des flux de données MIDI vers ou hors de la fonction audio.
Unité de mixage (MU - Mixer Unit)	Combine un certain nombre de canaux d'entrée logiques à un certain nombre de canaux de sortie logiques.
MUD	Acronyme de Mixer Unit descriptor (descripteur d'unité de mixage)
OTD	Acronyme d'Output Terminal descriptor (descripteur de terminal de sortie)
Broche de sortie	Connexion de sortie logique à une entité. Transporte une grappe unique.
Terminal de sortie (OT - Output Terminal)	Sortie pour les informations audio sortant de la fonction audio.
Boucle à verrouillage de phase	Système de commande matériel ou logiciel qui génère un signal de sortie dont la phase est liée à la phase d'un signal d'entrée. Généralement utilisé dans les applications de récupération d'horloge.
Unité de traitement (PU - Processing Unit)	Applique un processus prédéfini à un certain nombre de canaux d'entrée logiques.
PUD	Acronyme de Processing Unit descriptor (descripteur d'unité de traitement)

Réservé

Le mot clé "réservé" est utilisé pour indiquer les valeurs réservées d'un bit, d'un octet, d'un mot, d'un champ ou d'un code qui sont mises de côté pour une normalisation future. Leur utilisation et leur interprétation peuvent être spécifiées par de futures extensions à la présente spécification; sauf indication contraire, elles ne doivent pas être utilisées ou modifiées par la mise en œuvre d'un fournisseur. Un bit, un octet, un mot ou un champ réservé doit être défini sur zéro par l'émetteur et doit être ignoré par le destinataire.

RUD

Acronyme de Sampling Rate Converter Unit descriptor (descripteur d'unité de conversion du débit d'échantillonnage)

Unité de conversion du débit d'échantillonnage (RU - sampling Rate converter Unit)

Convertit le flux de données audio entrant, circulant à un débit d'échantillonnage donné, synchrone à une première horloge maître, en un flux de données audio sortant qui circule à un débit d'échantillonnage donné, synchrone à une deuxième horloge maître, en fréquence indépendante par rapport à la première horloge maître.

Unité de sélection (SU - Selector Unit)

Effectue un choix entre un certain nombre de grappes d'entrée.

SUD

Acronyme de Selector Unit descriptor (descripteur d'unité de sélection)

Terminal

Objet logique adressable au sein d'une fonction audio qui représente une connexion au monde extérieur de la fonction audio.

Unité

Objet logique adressable au sein d'une fonction audio qui représente une certaine sous-fonctionnalité audio.

XUD

Acronyme d'Extension Unit descriptor (descripteur d'unité d'extension).

2

VUE D'ENSEMBLE DE GESTION

L'USB est particulièrement bien adapté au transport audio, depuis les connexions vocales à faible fidélité jusqu'aux flux audio multicanaux de haute qualité. L'USB est devenu un connecteur omniprésent sur les PC modernes et est aujourd'hui bien compris par la plupart des consommateurs. Ainsi, il est devenu le connecteur de choix pour de nombreux périphériques et est en effet le connecteur audio numérique le plus simple et le plus répandu disponible aujourd'hui. Les consommateurs peuvent compter sur ce support pour répondre à tous leurs besoins audio actuels et futurs. De nombreuses applications, des communications au divertissement, en passant par l'enregistrement et la lecture de musique, peuvent profiter des fonctionnalités audio de l'USB.

En principe, une spécification de bus polyvalente comme l'USB fournit de nombreux moyens de propager et/ou de contrôler l'audio numérique. Pour l'industrie, cependant, il est très important de bien définir et normaliser les mécanismes de transport audio sur l'USB. L'interopérabilité ne peut être garantie que de cette manière parmi les nombreux dispositifs audio possibles sur l'USB. Les mécanismes de transport audio normalisés aident également à maintenir les pilotes logiciels aussi génériques que possible. La classe des dispositifs audio décrite dans le présent document satisfait à ces exigences. Il a été rédigé et révisé par des experts du domaine audio. Il convient que d'autres classes de dispositifs traitant de l'audio d'une manière ou d'une autre fassent référence au présent document dans le cadre de leur spécification d'interface audio.

Un problème essentiel de l'audio est la synchronisation des flux de données. En effet, les plus petits artefacts sont facilement détectés par l'oreille humaine. Par conséquent, un schéma de synchronisation robuste sur les transferts isochrones a été développé et incorporé dans la *Spécification USB* et dans la *Spécification USB 3.1*. La définition de classes de dispositifs audio respecte ces schémas de synchronisation pour transporter les données audio de manière fiable sur le bus.

Le présent document contient toutes les informations nécessaires pour qu'un concepteur construise un dispositif conforme à l'USB intégrant la fonctionnalité audio. Il spécifie les descripteurs normalisés et spécifiques à la classe qui doivent être présents dans chaque fonction audio USB. Il explique en outre l'utilisation des demandes spécifiques à la classe permettant un contrôle total de la fonction audio. Un certain nombre de formats de données prédéfinis sont répertoriés et entièrement documentés. Chaque format définit un mode de transport audio normalisé sur l'USB. Des dispositions ont été prises pour que les formats audio et les schémas de compression spécifiques au fournisseur puissent également être traités.

De nombreuses modifications introduites dans la présente version de la spécification USB pour les dispositifs audio ont été inspirées par la volonté d'utiliser l'audio USB dans les dispositifs portatifs modernes. Une attention particulière a été portée afin de rendre la classe des dispositifs audio plus écoénergétique en fournissant de nouveaux outils pour activer et désactiver de manière sélective des parties de la fonction audio, mais aussi en prenant en charge les transferts de données en mode en salves pour des temps de veille plus longs entre les transferts de données. En outre, la spécification prend en charge de nouveaux types de codecs et formats de données pour les applications audio consommateur, fournit de nombreuses clarifications par rapport à la spécification originale ainsi que des extensions pour la prise en charge de différentes modifications apportées à la spécification principale.

2.1

VUE D'ENSEMBLE DES PRINCIPALES DIFFERENCES ENTRE ADC V2.0 ET V3.0

La liste suivante n'est pas une liste exhaustive de toutes les modifications qui ont été introduites. Pour obtenir tous les informations, se référer à la spécification complète.

- Suppression de l'unité de traitement Dolby
- Suppression de la prise en charge de l'encodeur et du décodeur
- Suppression de la protection contre la copie (style S/PDIF)

- Suppression du champ bmAttributes (bit MaxPacketsOnly) du descripteur de point d'extrémité isochrone spécifique à la classe
- Suppression des formats de données audio de type II et de type II étendu
- Nouveaux domaines d'alimentation
- Nouvelle unité de traitement multifonction
- Introduction de formats de type III supplémentaires
- Nouveau descripteur de grappe
- Nouveau descripteur de terminal étendu
- Nouveau descripteur de connecteurs
- Modification de la topologie de tous les descripteurs spécifiques à la classe
- Nouveaux descripteurs de chaîne spécifiques à la classe
- Nouveaux descripteurs haute capacité
- Sources supplémentaires d'interruption
- Rétrocompatibilité à travers plusieurs descripteurs de configuration
- Prise en charge de LPM/L1
- Prise en charge obligatoire de BADD sur chaque dispositif conforme ADC3.0

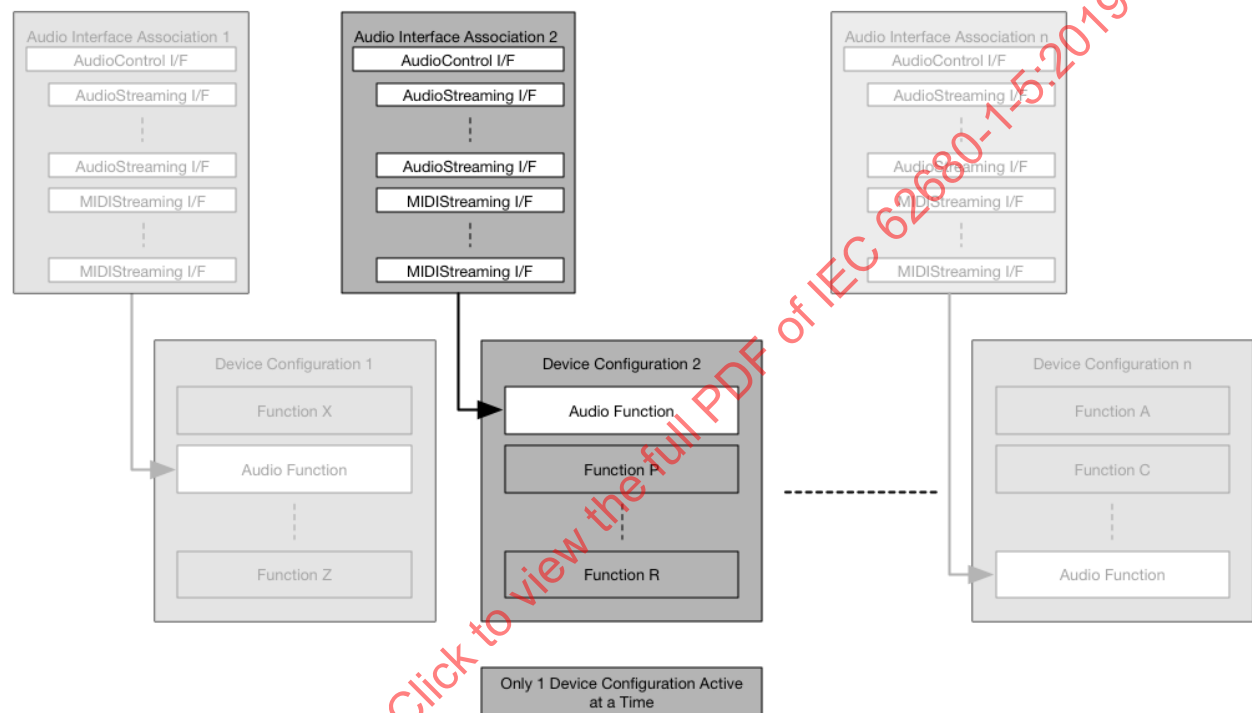
IECNORM.COM : Click to view the full PDF of IEC 62680-1-5:2019

3 CARACTERISTIQUES FONCTIONNELLES

3.1 INTRODUCTION

Dans de nombreux cas, la fonctionnalité audio n'existe pas en tant que dispositif autonome. Il s'agit d'une capacité qui, avec d'autres fonctions, constitue un dispositif "composite". Un lecteur DVD-ROM, qui peut intégrer la vidéo, l'audio, le stockage de données et le contrôle du transport, en est un parfait exemple. La fonction audio se situe donc au niveau de l'interface dans la hiérarchie des classes de dispositifs. La figure suivante fournit des détails.

Figure 3-1: Plusieurs configurations et leurs interfaces et associations d'interfaces audio



Anglais	Français
Audio Interface Association	Association d'interfaces audio
I/F	Interface
Device Configuration	Configuration de dispositif
Function	Fonction
Audio Function	Fonction audio
Only 1 Device Configuration Active at a Time	Une seule configuration de dispositif activée à la fois

Une fonction audio est vue comme une "boîte fermée" qui dispose d'interfaces très distinctes et bien définies avec le monde extérieur. Les fonctions audio sont adressables et accessibles par le biais d'une association d'interfaces audio (AIA). L'AIA regroupe toutes les interfaces USB qui, ensemble, permettent l'accès à la fonction audio à des fins de commande et de diffusion.

Un dispositif USB peut exprimer différentes AIA représentant le même matériel sous-jacent en utilisant plusieurs descripteurs de configuration de dispositif USB. Ces AIA peuvent être utilisées pour fournir un accès à la fonction audio à différents niveaux de conformité. Par exemple, une AIA peut fournir l'accès à la fonction audio en mode ADC 1.0, tandis qu'une autre AIA peut fournir l'accès à la fonction audio en mode ADC 3.0, comme défini par le présent document. Noter que la fonction audio peut présenter une topologie différente et même des fonctionnalités différentes selon l'accès par différentes AIA.

Comme une seule configuration peut être active à la fois, une seule AIA doit être active à la fois pour la même fonction audio. Il convient que le logiciel hôte choisisse la configuration la plus souhaitable et utilise la commande normalisée SET_CONFIGURATION pour choisir la configuration comprenant la ou les AIA les plus souhaitées (si plusieurs fonctions audio indépendantes résident dans le même dispositif USB).

Une AIA doit disposer d'une interface AudioControl unique et peut n'avoir aucune ou avoir plusieurs interfaces AudioStreaming et n'avoir aucune ou avoir plusieurs interfaces MIDIStreaming. L'interface AudioControl (AC) est utilisée pour accéder aux commandes audio de la fonction, tandis que les interfaces AudioStreaming (AS) sont utilisées pour transporter les flux audio vers et hors de la fonction audio. Les interfaces MIDIStreaming (MS) peuvent être utilisées pour transporter des flux de données MIDI vers et hors de la fonction audio.

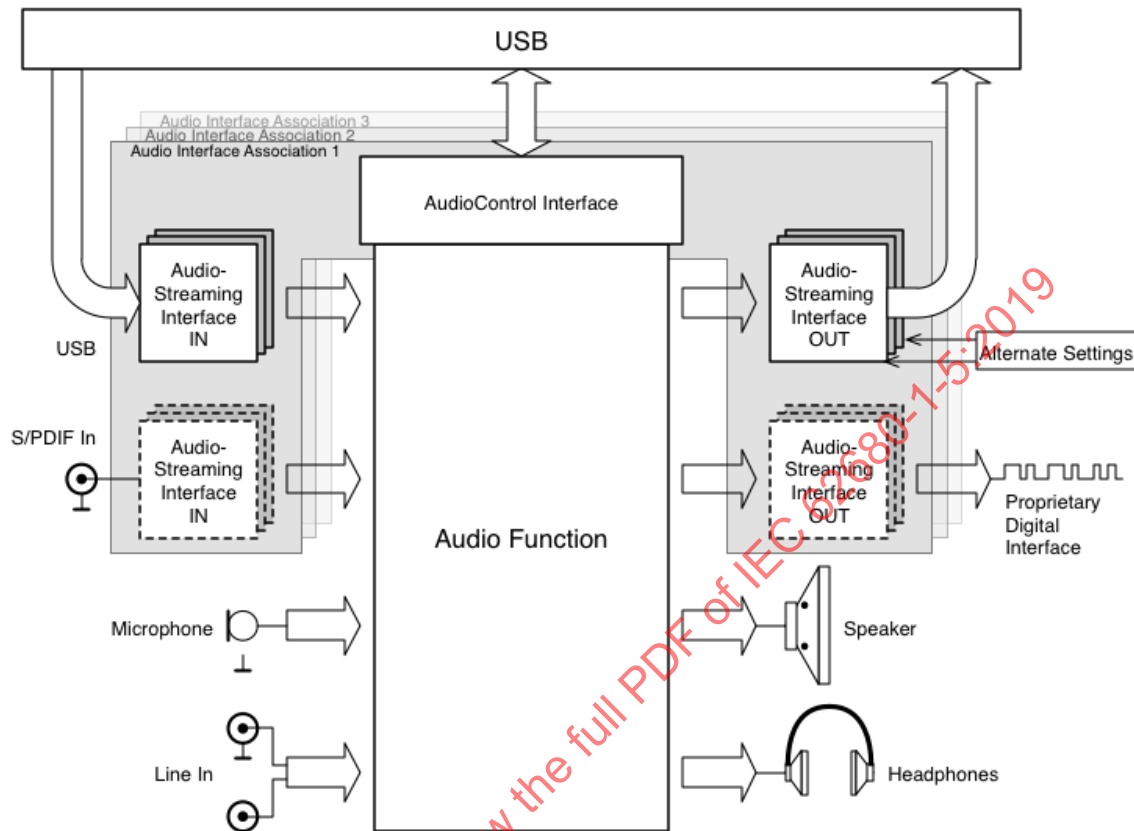
Note: Toutes les informations relatives au protocole MIDI sont regroupées dans un document distinct, *Universal Serial Bus Device Class Definition for MIDI Devices* (disponible en anglais seulement), qui est vu comme faisant partie de la présente spécification. Les interfaces MIDIStreaming et leurs spécificités ne seront donc plus mentionnées dans le reste du présent document.

Noter que les interfaces AudioStreaming ne représentent pas nécessairement toutes des flux audio USB. Tout flux audio, quelle que soit son origine ou sa destination, qui entre ou sort de la fonction audio, peut avoir une interface AudioStreaming associée, si cette interface exige que l'hôte ait accès à et/ou ait une visibilité sur certaines commandes ou certains descripteurs d'interfaces qui peuvent avoir un impact sur le fonctionnement de l'interface d'un point de vue USB.

Un dispositif peut avoir *plusieurs* fonctions audio *indépendantes* situées dans le même dispositif composite. Elles sont toutes accessibles au moyen de leur association d'interfaces audio active, qui doit être comprise dans la configuration USB actuellement choisie.

La figure suivante représente le concept d'une fonction audio et de ses interfaces associées:

Figure 3-2: Vue d'ensemble de la fonction audio



Anglais	Français
Audio Interface Association	Association d'interfaces audio
AudioControl Interface	Interface AudioControl
AudioStreaming Interface IN	Interface AudioStreaming IN
S/PDIF In	Entrée S/PDIF
Audio Function	Fonction audio
Line In	Entrée de ligne
AudioStreaming Interface OUT	Interface AudioStreaming OUT
Alternate Settings	Paramètres alternatifs
Proprietary Digital Interface	Interface numérique propriétaire
Speaker	Haut-parleur
Headphones	Casque

Toutes les fonctionnalités relatives au contrôle des paramètres qui influencent directement la perception audio (comme le volume) sont situées à l'intérieur du rectangle central et sont exclusivement commandées par l'interface AudioControl. Les aspects relatifs à la diffusion des communications vers ou à partir de la fonction audio sont gérés par des interfaces AudioStreaming distinctes, selon les besoins. Chaque flux audio USB doit être représenté par une interface AudioStreaming. Un flux audio non USB peut être représenté par une interface AudioStreaming si la transmission d'informations entre l'interface de ce flux audio et l'hôte est nécessaire. Toutes les données de commande liées spécifiquement au comportement de diffusion de l'interface sont transmises par le biais de l'interface AudioStreaming.

Une connexion S/PDIF physique à la fonction audio constitue un exemple type d'une interface AudioStreaming non USB. Bien que les données audio actuelles proviennent du monde extérieur (et non de l'USB), il peut être nécessaire de contrôler certains aspects de la connexion S/PDIF. Dans ce cas, la connexion S/PDIF est représentée par une interface AudioStreaming qui la rend adressable par USB.

Noter également que la connexion entre les interfaces AudioStreaming et la fonction audio n'est pas "solide". La raison en est que, lorsqu'il est vu de l'intérieur de la fonction audio, chaque flux audio entrant ou sortant de la fonction audio est représenté par un objet spécial, appelé terminal (voir plus loin). Le concept de terminal abstraît l'interface AudioStreaming réelle dans la fonction audio et fournit une vue logique sur la connexion plutôt qu'une vue physique. Cette abstraction permet de traiter les canaux audio au sein de la fonction audio comme des canaux audio "logiques" qui n'ont plus de caractéristiques physiques associées (analogique vs. numérique, format, débit d'échantillonnage, résolution en bits, etc.).

3.2 DEFINITION DE DISPOSITIF AUDIO DE BASE

La définition de dispositif audio de base (BADD) existe en tant que document distinct et est fondée sur et entièrement compatible avec la présente définition de classes de dispositifs audio (ADC 3.0). Ainsi, la BADD définit un sous-ensemble de fonctionnalités généralement présentes dans les casques, les microphones et les micro-casques, en utilisant les fonctions et les outils fournis par l'ADC, mais elle supprime toutes les options admises par l'ADC et décrit rigoureusement la manière dont certaines fonctions doivent être mises en œuvre. Cela permet aux pilotes hôtes simples de savoir exactement à quoi s'attendre lorsqu'ils rencontrent un dispositif conforme BADD, réduisant considérablement le travail qu'un pilote a besoin d'effectuer pour analyser les descripteurs du dispositif BADD et interagir avec ce dispositif.

3.3 RETROCOMPATIBILITE

La présente version 3.0 de la *définition de classes de dispositifs audio* (ADC 3.0) en tant que telle n'est rétrocompatible avec aucune des versions précédentes de cette même spécification.

Toutefois, afin d'assurer la plus grande interopérabilité possible entre les nouveaux dispositifs ADC 3.0 et les anciens hôtes ADC 1.0 ou ADC 2.0, une exigence de la présente spécification est que le dispositif conforme ADC 3.0 doive satisfaire à ce qui suit:

- Le premier descripteur de configuration USB (indice 0) exposé par le dispositif doit contenir une association d'interfaces audio (AIA) conforme à ADC 1.0 ou ADC 2.0. En d'autres termes, si le logiciel hôte choisit la première configuration de dispositif indexée, il doit exposer une fonction audio conforme à ADC 1.0 ou ADC 2.0. Il convient que cette exigence permette à la fonction audio d'interopérer avec les hôtes existants exactement de la même manière que les fonctions ADC 1.0 ou ADC 2.0 héritées.
- Si la fonction audio contient au moins une interface de diffusion audio USB, le dispositif doit avoir un autre descripteur de configuration qui expose une AIA conforme BADD 3.0. Un dispositif ne doit avoir qu'une seule AIA de ce type. Toutefois, il peut être nécessaire de dupliquer cette même AIA conforme BADD dans plusieurs descripteurs de configuration. Cette exigence garantit que le dispositif interopérera avec les nouveaux hôtes ADC 3.0 qui prennent en charge la spécification BADD 3.0. Pour activer ce mode de fonctionnement pour la fonction audio, il est nécessaire que l'hôte choisisse une configuration de dispositif comprenant l'AIA conforme BADD 3.0.
- Le dispositif peut avoir une ou plusieurs configurations qui exposent une ou plusieurs AIA qui doivent être conformes à ADC 3.0. De cette façon, la fonction audio peut fournir des fonctionnalités au-delà de ce qui est disponible en mode de fonctionnement BADD 3.0. Pour activer ce mode de fonctionnement, il est nécessaire que l'hôte choisisse explicitement une configuration qui comprend la ou les AIA souhaitées.

Il est fortement recommandé que TOUS les hôtes ADC 3.0 prennent en charge la spécification BADD 3.0.

3.4 ASSOCIATION D'INTERFACES AUDIO (AIA) ET DESCRIPTEUR D'ASSOCIATION D'INTERFACES

Sur l'USB, une fonction audio est entièrement définie par ses interfaces. Une fonction audio possède une ou plusieurs associations d'interfaces audio (AIA) qui lui donnent accès à différents niveaux de conformité. Une seule AIA peut être active à la fois. Chaque AIA doit avoir une interface AudioControl et peut n'avoir aucune ou avoir plusieurs interfaces AudioStreaming.

Le mécanisme d'association d'interfaces USB normalisé est utilisé pour décrire l'association d'interfaces audio, c'est-à-dire pour lier ces interfaces ensemble. L'association d'interfaces est exprimée au moyen du descripteur d'association d'interfaces (IAD) USB normalisé. Chaque descripteur d'association d'interfaces dispose d'un champ **bFunctionClass**, d'un champ **bFunctionSubClass** et d'un champ **bFunctionProtocol** qui, ensemble, identifient la fonction représentée par l'association. Les alinéas suivants définissent ces champs pour la classe des dispositifs audio.

3.4.1 CLASSE DE FONCTION AUDIO

La classe de fonction audio est contenue dans le champ **bFunctionClass** d'un descripteur d'association d'interfaces normalisé. La présente spécification exige que le code de classe de fonction soit le même que le code de classe d'interface audio.

Le code de classe de fonction audio est attribué par l'USB-IF. Pour plus d'informations, voir Annexe A.1, "Code de classe de fonction audio".

3.4.2 SOUS-CLASSE DE FONCTION AUDIO

La sous-classe de fonction audio est contenue dans le champ **bFunctionSubClass** d'un descripteur d'association d'interfaces normalisé. Toute AIA exposée par un dispositif conforme à la présente spécification doit utiliser ce champ pour définir plus précisément le dispositif. Ce champ indique si l'AIA est une mise en œuvre intégrale de la spécification ADC 3.0 ou si elle est conforme à l'un des profils définis par la BADD.

Les codes attribués peuvent être consultés en A.2, "Codes de sous-classes de fonction audio" de la présente spécification. Tous les autres codes de sous-classe sont inutilisés et réservés par la présente spécification à une utilisation future.

3.4.3 PROTOCOLE DE FONCTION AUDIO

Le protocole de fonction audio est contenu dans le champ **bFunctionProtocol** d'un descripteur d'association d'interfaces normalisé. Le code de protocole de fonction est utilisé pour refléter le niveau de conformité de la fonction audio afin que le logiciel d'énumération puisse décider quelle version de pilote il est nécessaire d'instancier. La présente spécification exige que le code de protocole de fonction soit identique au code de protocole d'interface audio (AudioControl et AudioStreaming).

Les codes de protocole attribués peuvent être consultés en A.3, "Codes de protocoles de fonction audio" de la présente spécification. Tous les autres codes de protocole sont inutilisés et réservés par la présente spécification à une utilisation future.

3.5 CLASSE D'INTERFACE AUDIO

Le code de classe d'interface audio est contenu dans le champ **bInterfaceClass** du descripteur d'interface normalisé de toute interface comprise dans une AIA. Toutes les interfaces USB qui fournissent des fonctionnalités à travers la présente spécification utilisent le même code de classe d'interface audio.

Le code de classe Interface audio est attribué par l'USB. Pour plus d'informations, voir Annexe A.4, "Code de classe d'interface audio".

3.6 SOUS-CLASSE D'INTERFACE AUDIO

Le code de sous-classe d'interface audio est contenu dans le champ **bInterfaceSubClass** du descripteur d'interface normalisé de toute interface comprise dans une AIA. Toutes les interfaces USB qui fournissent des fonctionnalités à travers la présente spécification, ainsi que toutes les interfaces qui fournissent des fonctionnalités de diffusion MIDI, doivent utiliser l'un des codes de sous-classe d'interface audio suivants, fournis par la présente spécification:

- Sous-classe d'interface AudioControl
- Sous-classe d'interface AudioStreaming
- Sous-classe d'interface MIDIStreaming

Les codes attribués peuvent être consultés en A.5, "Codes de sous-classes d'interface audio" de la présente spécification. Tous les autres codes de sous-classe sont inutilisés et réservés par la présente spécification à une utilisation future.

3.7 PROTOCOLE D'INTERFACE AUDIO

Le code de protocole d'interface audio est contenu dans le champ **bInterfaceProtocol** du descripteur d'interface normalisé de toute interface comprise dans une AIA. Le code de protocole d'interface est utilisé pour refléter le niveau de conformité de la fonction audio.

Toutes les fonctions audio conformes à la présente spécification ADC 3.0 doivent utiliser le code de protocole d'interface IP_VERSION_03_00. Les codes attribués peuvent être consultés en A.6, "Codes de protocoles d'interface audio" de la présente spécification. Tous les autres codes de protocole sont inutilisés et réservés par la présente spécification à une utilisation future.

3.8 CATEGORIE DE FONCTION AUDIO

Le code de catégorie de fonction audio est contenu dans le descripteur d'interface de commande audio spécifique à la classe défini par la présente spécification. Ce code indique l'utilisation principale prévue pour la fonction audio. Les catégories de fonctions suivantes sont actuellement définies dans la présente spécification:

- Haut-parleur de bureau: Un ou plusieurs haut-parleurs installés dans un environnement de petite taille pour fournir de l'audio destiné principalement à une personne.
- Cinéma maison: Plusieurs haut-parleurs installés dans un environnement de taille moyenne pour fournir des niveaux audio nettement plus élevés qu'une configuration de haut-parleur de bureau et destinés à être clairement audibles par plusieurs personnes.
- Microphone: Dispositif configuré pour enregistrer de l'audio en provenance de sources sonores.
- Micro-casque: Dispositif comprenant au moins un haut-parleur et au moins un microphone conçu pour être porté ou tenu par un utilisateur afin de fournir des capacités personnelles de lecture audio et d'entrée vocale.
- Téléphone: Micro-casque ou dispositif de type combiné qui se connecte également à un système téléphonique (par exemple POT, PBX, VoIP) capable de passer et de recevoir des appels téléphoniques.
- Convertisseur: Dispositif permettant la conversion de l'audio à partir d'un format électrique ou optique vers un autre format électrique ou optique et/ou la conversion de données audio à partir d'un format de codage vers un autre (par exemple AC-3 vers MIC, etc.).
- Enregistreur vocal/sonore: Dispositif comprenant au moins un microphone et au moins un haut-parleur conçu pour fonctionner, au moins de temps en temps, indépendamment de l'hôte afin d'enregistrer et de stocker des sources sonores et de lire le contenu enregistré.
- Boîtier E/S: Dispositif conçu pour fournir une ou plusieurs entrées et sorties électriques et optiques, éventuellement différentes, pour permettre la connexion à d'autres dispositifs.

- Instrument de musique: Instrument de musique, par exemple piano, guitare, synthétiseur, boîte à rythme, etc.
- Pro-Audio: Dispositif peu utilisé par les consommateurs d'audio, par exemple matériel de montage, matériel d'enregistrement multipiste, etc.
- Audio/Vidéo: L'audio d'un dispositif qui fournit également de la vidéo simultanée, où il est attendu que l'audio soit étroitement couplé à la vidéo, par exemple un caméscope, un lecteur DVD, une télévision, etc.
- Panneau de commande: Dispositif utilisé pour contrôler le flux audio au moyen d'un système de dispositifs audio, comme un panneau de mixage.
- Autre: Tout dispositif dont le but principal est suffisamment différent des descriptions ci-dessus pour être vu comme une forme complètement différente de dispositif.

Les codes attribués peuvent être consultés en A.7, "Codes de catégories de fonction audio" de la présente spécification. Tous les autres codes de catégorie sont inutilisés et réservés par la présente spécification à une utilisation future.

3.9 DOMAINES D'HORLOGE

Un domaine d'horloge est défini comme une zone dans laquelle toutes les horloges d'échantillonnage sont dérivées de la même horloge maître. Par conséquent, dans le même domaine de l'horloge, toutes les horloges d'échantillonnage sont synchrones et leur relation temporelle est constante. Cependant, les horloges d'échantillonnage peuvent présenter des fréquences d'échantillonnage différentes. L'horloge maître peut être générée de différentes façons. Un cristal interne peut constituer l'horloge maître; le début de trame (SOF - start of frame) USB peut être utilisé ou encore une horloge fournie de manière externe peut servir d'horloge maître.

Plusieurs domaines d'horloge différents peuvent exister dans la même fonction audio.

3.10 DOMAINES D'ALIMENTATION

Un domaine d'alimentation est défini comme une zone de la fonction audio qui regroupe une ou plusieurs entités et permet à l'hôte de contrôler les niveaux de consommation d'énergie de ce domaine. De cette façon, l'hôte peut potentiellement faire passer des parties de la fonction audio à des états d'alimentation inférieurs lorsque ces parties ne sont pas en cours d'utilisation, ce qui entraîne une diminution globale de la consommation d'énergie. Par exemple, un micro-casque USB peut disposer de deux domaines d'alimentation distincts, l'un pour la fonctionnalité liée à la sortie (casque) et l'autre pour la fonctionnalité liée à l'entrée (microphones). Lorsque le micro-casque est utilisé pour simplement écouter de la musique, la fonctionnalité liée à l'entrée peut être temporairement commutée vers un état de puissance faible (voire complètement désactivée) afin d'économiser de l'énergie. Cela est particulièrement important si le micro-casque est utilisé en conjonction avec un dispositif alimenté par batterie, comme un téléphone mobile ou un lecteur MP3, où l'allongement de la durée de vie de la batterie est important.

Plusieurs domaines d'alimentation différents peuvent exister dans la même fonction audio. Les domaines d'alimentation sont identifiés par un ID de domaine d'alimentation unique au sein de la fonction audio.

3.11 TYPES DE SYNCHRONISATION AUDIO

Chaque point d'extrémité audio isochrone utilisé dans une interface AudioStreaming appartient à un type de synchronisation défini à la Section 5 de la *Spécification USB*. Les sections suivantes décrivent brièvement les types de synchronisation possibles.

3.11.1 ASYNCHRONE

Les points d'extrémité audio isochrones asynchrones produisent ou consomment des données à un débit qui est verrouillé soit sur une horloge externe à l'USB, soit sur une horloge interne en fréquence indépendante. Ces points d'extrémité asynchrones ne sont pas synchronisés à un début de trame (SOF) ni à une autre horloge du domaine USB.

3.11.2 SYNCHRONE

Le système d'horloge des points d'extrémité audio isochrones synchrones peut être contrôlé de manière externe à travers la synchronisation du SOF ou de l'intervalle de bus. Un tel point d'extrémité doit verrouiller son horloge d'échantillonnage sur la coche SOF ou le début d'un nouvel intervalle de bus.

3.11.3 ADAPTATIVE

Les points d'extrémité audio isochrones adaptatifs sont capables de jouer le rôle de source ou de destinataire de données avec n'importe quel débit de leur plage de fonctionnement. Cela implique que ces points d'extrémité doivent exécuter un processus interne leur permettant de faire correspondre leur débit de données naturel au débit de données imposé à leur interface.

3.11.4 IMPLICATIONS DES DIFFERENTS TYPES DE SYNCHRONISATION

Les sections suivantes fournissent des informations relatives à l'impact des différents types de synchronisation sur la mise en œuvre du dispositif et sur celle de l'hôte. Plusieurs scénarios sont pris en compte et les implications côté hôte et côté dispositif sont répertoriées pour chaque type de synchronisation.

3.11.4.1 POINT D'EXTREMITE DESTINATAIRE A SENS UNIQUE

Un haut-parleur USB, mettant en œuvre un point d'extrémité destinataire qui reçoit des données audio de diffusion en provenance de l'hôte, constitue un exemple type de ce scénario.

- Asynchrone
 - **Hôte:** Il est nécessaire que le pilote hôte soit capable de gérer un point d'extrémité de rétroaction explicite. À partir des données de rétroaction, l'hôte décide du nombre d'échantillons à envoyer sur le point d'extrémité de diffusion des données dans les intervalles de bus suivants.
 - **Dispositif:** Le dispositif possède sa propre horloge d'échantillonnage audio locale en fréquence indépendante, qui détermine le nombre d'échantillons consommés par le dispositif à chaque intervalle de bus. Le dispositif doit mettre en œuvre un point d'extrémité de rétroaction explicite ainsi que la logique nécessaire pour fournir les valeurs de rétroaction correctes à renvoyer à l'hôte sur ce point d'extrémité. L'avantage de ce mode de fonctionnement est qu'il est assez facile de générer une horloge d'échantillonnage audio robuste, stable, sans gigue et de haute qualité (dérivée d'une horloge maître à base de cristal, par exemple).
- Synchrone
 - **Hôte:** Il est nécessaire que l'hôte envoie un nombre d'octets connu pour chaque paquet à destination du dispositif. Il peut être nécessaire que l'hôte génère un modèle (fixe) d'échantillons audio pour atteindre le débit d'échantillonnage souhaité. Par exemple, pour générer un débit d'échantillonnage de 44,1 kHz dans une mise en œuvre à vitesse intégrale, il est nécessaire que l'hôte envoie un modèle de répétition de neuf paquets contenant 44 échantillons audio, suivis d'un paquet contenant 45 échantillons audio.

- **Dispositif:** Ce type de synchronisation exige que le dispositif mette en œuvre une PLL d'horloge audio ou une ASRC.
- Adaptive
 - **Hôte:** L'hôte peut utiliser n'importe quel moyen ou méthode pour déterminer le nombre d'échantillons à transmettre par intervalle de bus. Un fonctionnement efficace en tant que source "synchrone au SOF" constitue une approche facile, mais ce n'est pas la seule admise par la spécification USB.
 - **Dispositif:** Ce type de synchronisation exige que le dispositif mette en œuvre soit une PLL d'horloge audio, soit une ASRC afin de s'adapter au nombre moyen d'échantillons arrivant au cours d'une période donnée.

3.11.4.2 POINT D'EXTREMITÉ SOURCE A SENS UNIQUE

Un microphone USB, mettant en œuvre un point d'extrémité source qui envoie des données audio de diffusion à l'hôte, constitue un exemple type de ce scénario.

- Asynchrone
 - **Hôte:** Il est nécessaire que l'hôte comme un destinataire adaptatif. Selon la conception du système de l'hôte, cela peut exiger une ASRC ou une PLL d'horloge audio du côté hôte.
 - **Dispositif:** Le dispositif possède sa propre horloge d'échantillonnage audio locale en fréquence indépendante, qui détermine le nombre d'échantillons produits par le dispositif à chaque intervalle de bus. L'avantage de ce mode de fonctionnement est qu'il est assez facile de générer une horloge d'échantillonnage audio robuste, stable, sans gigue et de haute qualité (dérivée d'une horloge maître à base de cristal, par exemple).
- Synchrone
 - **Hôte:** L'hôte reçoit un nombre connu d'échantillons audio dans chaque paquet en provenance du dispositif. Il convient que l'hôte s'attende à ce qu'un modèle fixe d'échantillons audio atteigne le débit d'échantillonnage souhaité.
 - **Dispositif:** Ce type de synchronisation exige que le dispositif mette en œuvre une PLL d'horloge audio afin de se verrouiller sur le SOF ou sur le début de l'intervalle de bus et de générer directement une horloge d'échantillonnage audio de haute qualité, ou qu'il utilise une ASRC en tant que pont entre les domaines d'horloge local et USB. Le dispositif doit générer un modèle de taille de paquet fixe, comme décrit dans le document *USB Audio Data Formats Definition* (disponible en anglais seulement).
- Adaptive
 - **Hôte:** Ce scénario exige la mise en œuvre d'un point d'extrémité de prédiction OUT dans le dispositif, permettant à l'hôte d'informer le dispositif du nombre d'échantillons par intervalle de bus à envoyer à l'hôte sur le point d'extrémité de données de diffusion IN.
 - **Dispositif:** Ce type de synchronisation exige que le dispositif mette en œuvre une PLL d'horloge audio ou une ASRC afin de s'adapter au débit d'échantillonnage communiqué par l'hôte à travers le point d'extrémité de prédiction OUT.

3.11.4.3 POINTS D'EXTREMITÉ SOURCE ET DESTINATAIRE

Un micro-casque USB, mettant en œuvre à la fois un point d'extrémité destinataire qui reçoit des données audio de diffusion en provenance de l'hôte et un point d'extrémité source qui envoie des données audio de diffusion à l'hôte, constitue un exemple type de ce scénario.

En général, les points d'extrémité peuvent être traités de manière indépendante, en utilisant les mêmes règles et lignes directrices que celles énoncées ci-dessus. Cela inclut la possibilité pour le dispositif de prendre en charge des horloges distinctes sur chaque point d'extrémité.

Toutefois, si le dispositif possède sa propre horloge et que les deux flux de données partagent cette horloge, les deux cas particuliers suivants peuvent être utilisés pour permettre une rétroaction implicite:

- Les deux points d'extrémité sont asynchrones. Dans ce cas, le débit de données qui apparaît sur le point d'extrémité source peut être utilisé par l'hôte comme rétroaction implicite pour ajuster le débit de données transmis au point d'extrémité destinataire.
- Les deux points d'extrémité sont adaptatifs. Dans ce cas, le débit de données qui apparaît sur le point d'extrémité destinataire peut être utilisé par le dispositif pour ajuster le débit de données transmis par le point d'extrémité source.

Note: L'utilisation de l'un ou l'autre de ces deux mécanismes de rétroaction implicite empêche un dispositif ADC 3.0 d'utiliser les domaines d'alimentation pour désactiver la source ou le destinataire, les données devant continuer à circuler dans les deux sens pour fournir les informations de rétroaction/de prédiction.

3.12 SYNCHRONISATION ENTRE LES CANAUX

Un problème important posé par le traitement de l'audio, et en particulier de l'audio 3D, est la relation de phase entre différents canaux audio physiques. En effet, la position spatiale virtuelle d'une source audio est directement liée aux différences de phase qui sont appliquées aux différents canaux audio physiques utilisés pour reproduire la source audio et influencée par celles-ci. Par conséquent, il est impératif que les fonctions audio USB respectent la relation de phase entre tous les canaux audio associés. Cependant, la responsabilité du maintien de la relation de phase est partagée entre le logiciel hôte USB, le matériel et tous les dispositifs périphériques ou fonctions audio.

3.13 TOPOLOGIE DE FONCTION AUDIO

Pour pouvoir manipuler les propriétés physiques d'une fonction audio, ses fonctionnalités doivent être divisées en entités adressables. Deux types de telles entités génériques sont identifiés et appelés unités et terminaux. Un type spécial d'entité est en outre défini. Ces entités sont appelées entités d'horloge et sont utilisées pour décrire et manipuler les signaux d'horloge au sein de la fonction audio.

Les unités fournissent les blocs de construction de base pour décrire entièrement la plupart des fonctions audio. Les fonctions audio se construisent en connectant plusieurs de ces unités ensemble. Une unité possède une ou plusieurs broches d'entrée et une broche de sortie unique, chaque broche représentant une grappe de canaux audio logiques au sein de la fonction audio (voir Section 3.13.1, "Grappe"). Les unités sont reliées entre elles en connectant leurs broches d'E/de S selon la topologie exigée. Noter qu'il est parfaitement admis de connecter la broche de sortie d'une entité à plusieurs broches d'entrée résidant sur différentes autres entités, créant ainsi une connexion de type une-à-plusieurs.

En outre, le concept de terminal est introduit. Il existe deux types de terminaux. Un terminal d'entrée (IT) est une entité qui représente un point de départ pour les canaux audio au sein de la fonction audio. Un terminal de sortie (OT) représente un point de terminaison pour les canaux audio au sein de la fonction audio. Du point de vue de la fonction Audio, un point d'extrémité USB est un exemple type de terminal d'entrée ou de sortie. Soit il fournit des flux de données à la fonction audio (IT), soit il consomme des flux de données en provenance de la fonction audio (OT). De même, un convertisseur numérique-analogique intégré à la fonction audio est représenté comme un terminal de sortie dans le modèle de la fonction audio. La connexion au terminal s'effectue par l'intermédiaire de sa broche d'entrée ou de sortie unique.

Les broches d'entrée d'une unité sont numérotées à partir de un jusqu'au nombre total de broches d'entrée sur l'unité. Le numéro d'une broche de sortie est toujours un. Les terminaux d'entrée ne possèdent qu'une seule broche de sortie et son numéro est toujours un. Les terminaux de sortie ne possèdent qu'une seule broche d'entrée et elle est toujours numérotée à un.

Les informations traversant les broches d'E/de S ne sont pas nécessairement de nature numérique. Il est parfaitement possible d'utiliser le modèle d'unité pour décrire des fonctions audio entièrement analogiques ou même hybrides. Le simple fait que les broches d'E/de S soient connectées ensemble constitue une garantie (par construction) que le protocole et le format utilisés sur ces connexions (analogiques ou numériques) sont compatibles aux deux extrémités.

Chaque unité de la fonction audio est entièrement décrite par son descripteur d'unité (UD) associé. Le descripteur d'unité contient tous les champs nécessaires pour identifier et décrire l'unité. De même, il existe un descripteur de terminal (TD) pour chaque terminal de la fonction audio. En outre, ces descripteurs fournissent toutes les informations nécessaires concernant la topologie de la fonction audio. Ils décrivent pleinement la façon dont les terminaux et les unités sont interconnectés.

La présente spécification décrit les types d'unités et de terminaux normalisés suivants, qui sont jugés adéquats pour représenter la plupart des fonctions audio:

- Terminal d'entrée (IT)
- Terminal de sortie (OT)
- Unité de mixage (MU)
- Unité de sélection (SU)
- Unité de fonction (FU)
- Unité de conversion du débit d'échantillonnage (RU)
- Unité d'effet (EU)
- Unité de traitement (PU)
- Unité d'extension (XU)

En plus des unités et des terminaux, le concept d'entité d'horloge est introduit. Trois types d'entités d'horloge sont définis par la présente spécification:

- Source d'horloge (CS)
- Sélecteur d'horloge (CX)
- Multiplicateur d'horloge (CM)

Une source d'horloge fournit une certaine fréquence d'horloge d'échantillonnage à toute ou partie de la fonction audio. Une source d'horloge peut représenter un générateur de fréquence d'échantillonnage interne, mais elle peut aussi représenter une entrée de signal d'horloge d'échantillonnage externe à la fonction audio.

Une source d'horloge dispose d'une broche de sortie d'horloge unique transportant le signal d'horloge d'échantillonnage, représentée par la source d'horloge. Le numéro d'une broche de sortie d'horloge est toujours un.

Un sélecteur d'horloge est utilisé pour effectuer un choix entre plusieurs signaux d'horloge d'échantillonnage pouvant être disponibles au sein d'une fonction audio. Il possède plusieurs broches d'entrée d'horloge et une broche de sortie d'horloge unique. Les broches d'entrée d'horloge sont numérotées à partir de un jusqu'au nombre total de broches d'entrée d'horloge sur le sélecteur d'horloge. Le numéro d'une broche de sortie d'horloge est toujours un.

Un multiplicateur d'horloge est utilisé pour dériver un nouveau signal d'horloge avec une fréquence différente à partir du signal d'horloge au niveau de sa broche d'entrée d'horloge unique. Pour ce faire, il multiplie cette fréquence de signal d'horloge par un numérateur P, puis la divise par un dénominateur Q. Le nouveau signal d'horloge est garanti synchrone au signal d'horloge en entrée. Un multiplicateur d'horloge possède une broche d'entrée et une broche de sortie et leurs numéros sont toujours un. Une mise en œuvre donnée peut choisir de rendre les valeurs P et Q programmables, bien que la plupart des fonctions audio exposent les commandes P et Q en lecture seule et indiquent toute modification de ces valeurs à travers le mécanisme d'interruption. De telles modifications peuvent être nécessaires lorsque la fonction audio détecte une variation du débit d'échantillonnage natif du contenu codé entrant, par exemple.

En utilisant une combinaison d'entités de source d'horloge, sélecteur d'horloge et multiplicateur d'horloge, les systèmes d'horloge les plus complexes peuvent être représentés et exposés au logiciel hôte.

Les broches d'entrée et de sortie d'horloge sont fondamentalement différentes des broches d'entrée et de sortie définies pour les unités et les terminaux. Les broches d'horloge ne transportent que des signaux d'horloge et ne peuvent donc pas être connectées aux broches d'entrée et de sortie d'une unité ou d'un terminal. Elles ne sont utilisées que pour exprimer la topologie des circuits d'horloge.

Chaque terminal d'entrée et de sortie possède une broche d'entrée d'horloge unique connectée à une broche de sortie d'horloge d'une entité d'horloge. Le signal d'horloge transporté par cette broche de sortie d'horloge détermine à quelle fréquence d'échantillonnage le matériel représenté par le terminal fonctionne.

Chaque unité de conversion du débit d'échantillonnage possède deux broches d'entrée d'horloge qui sont généralement connectées aux broches de sortie d'horloge de deux entités d'horloge différentes. Les signaux d'horloge transportés par ces broches de sortie d'horloge déterminent les fréquences d'échantillonnage entre lesquelles l'unité de conversion du débit d'échantillonnage effectue la conversion.

Chaque entité d'horloge est décrite par un descripteur d'entité d'horloge (CED). Le descripteur d'entité d'horloge contient tous les champs nécessaires pour identifier et décrire l'entité d'horloge.

Les descripteurs sont décrits plus précisément à la Section 4, "Descripteurs" du présent document.

L'ensemble des descripteurs d'unité, des descripteurs de terminal et des descripteurs d'entité d'horloge fournit une description complète de la fonction audio à l'hôte. Ces informations sont généralement récupérées à partir du dispositif au moment de l'énumération. En analysant les descripteurs, il convient qu'un pilote audio générique soit capable de contrôler pleinement la fonction audio, à l'exception des fonctionnalités représentées par les unités d'extension. Celles-ci exigent des extensions spécifiques au fournisseur pour le pilote de classe audio.

Note importante:

L'ensemble complet des descripteurs de fonction audio fournit uniquement une description initiale statique de la fonction audio. Lors du fonctionnement, un certain nombre d'événements peuvent se produire, forçant la fonction audio à modifier son état. Le logiciel hôte doit être informé de ces modifications pour rester "synchronisé" avec la fonction audio à tout moment. Un mécanisme d'interruption complet est en place pour signaler tout changement d'état au logiciel hôte.

La Figure 3-3, "Au sein de la fonction audio" représente les concepts définis ci-dessus. En utilisant les symboles iconiques définis plus loin, elle décrit une fonction audio hypothétique comprenant 15 entités: trois terminaux d'entrée, cinq unités, trois terminaux de sortie, deux sources d'horloge, un sélecteur d'horloge et un multiplicateur d'horloge. Chaque entité possède son ID unique (de 1 à 15) et son descripteur décrivant pleinement la fonctionnalité de l'entité, ainsi que la façon dont cette entité particulière est connectée à la topologie globale de la fonction audio.

Le terminal d'entrée 1 (IT 1) est la représentation d'un point d'extrémité USB OUT utilisé pour diffuser l'audio en provenance de l'hôte vers le dispositif audio. IT 2 est la représentation d'un connecteur d'entrée de ligne analogique sur le dispositif audio tandis qu'IT 3 est un connecteur d'entrée de microphone analogique sur le dispositif audio. L'unité de sélection 4 (SU 4) choisit entre l'audio provenant de l'hôte et l'audio présent sur le connecteur d'entrée de ligne. L'unité de fonction 5 (FU 5) est ensuite utilisée pour manipuler l'audio (Volume, Grave, Aigu...) avant de le présenter au terminal de sortie 9 (OT 9). OT 9 est la représentation d'une prise de sortie de casque sur le dispositif audio.

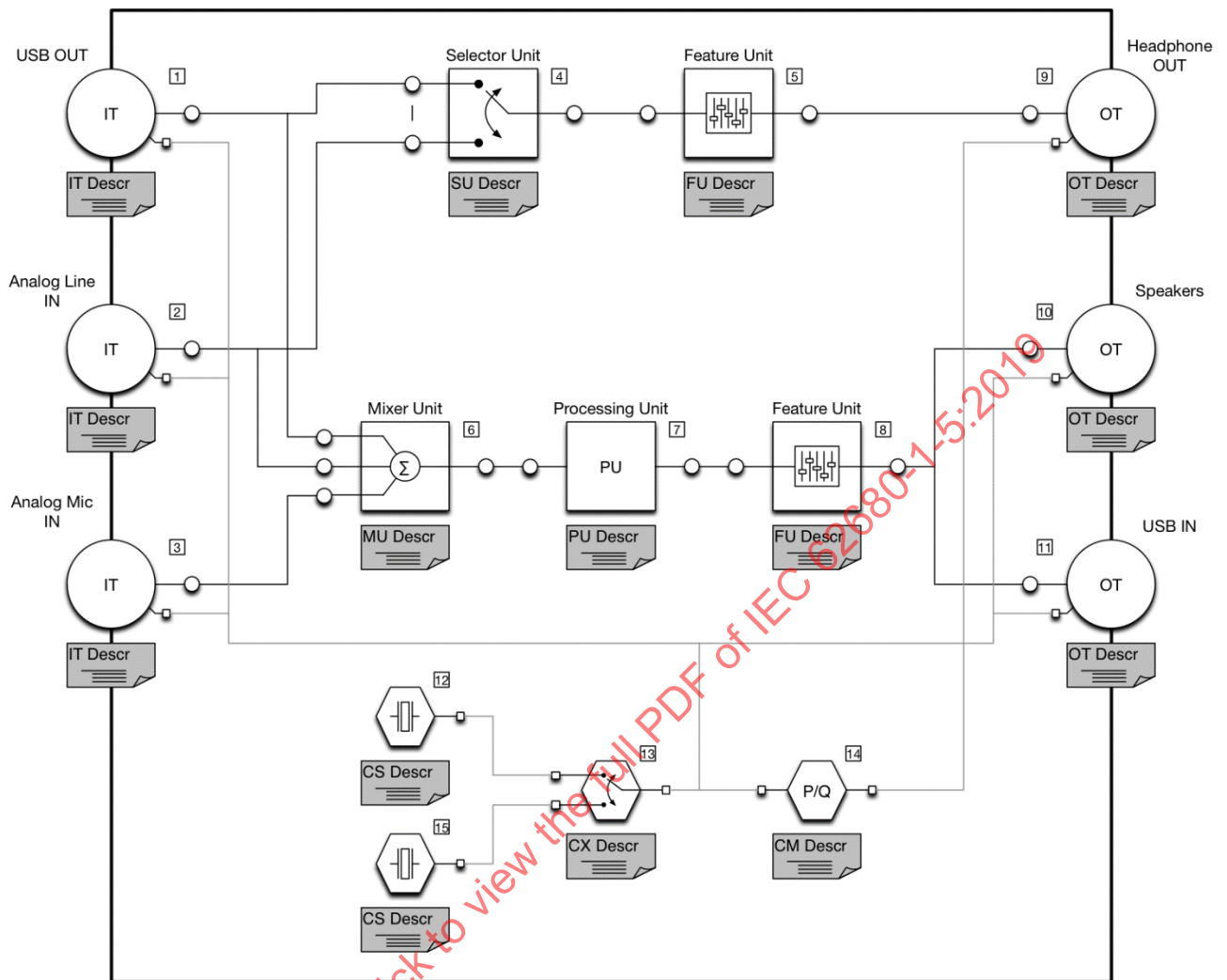
Dans le même temps, les trois sources d'entrée (USB OUT, Line-In et Mic-In) sont connectées à une unité de mixage 6 (MU 6) qui combine efficacement les trois sources. La sortie de l'unité de mixage est ensuite introduite dans une unité de traitement 7 (PU 7) qui exécute un ou plusieurs algorithmes de traitement audio sur la combinaison. Le résultat est à son tour envoyé à FU 8, où quelques ajustements finaux sont apportés à l'audio (volume...). FU 8 est connectée à OT 10 et OT 11. OT 10 représente les haut-parleurs intégrés au dispositif audio et OT 11 représente un point d'extrémité USB IN utilisé pour envoyer l'audio traité à l'hôte à des fins d'enregistrement.

La source d'horloge 12 (CS 12) représente un générateur de fréquence d'échantillonnage interne, fonctionnant à 96 kHz, par exemple. La source d'horloge 15 (CS 15) est la représentation d'une entrée d'horloge d'échantillonnage maître externe qui peut être utilisée pour synchroniser le dispositif à une source externe. Le sélecteur d'horloge 13 (CS 13) permet de choisir entre les deux sources d'horloge disponibles. La sortie de CS 13 fournit une fréquence d'échantillonnage de 96 kHz à IT 1, IT 2, IT 3, OT 10 et OT 11. Le multiplicateur d'horloge CM 14 multiplie ensuite ce signal d'horloge par 0,5, fournissant une fréquence d'échantillonnage de 48 kHz à OT 9 pour le pilotage du casque. Étant donné que toutes les fréquences d'échantillonnage utilisées au sein la fonction audio sont à tout moment dérivées d'une horloge maître unique (interne ou externe), tous les flux audio de la fonction audio sont synchrones.

Les descripteurs, associés à chaque entité, indiquent clairement à l'hôte la nature exacte de chaque entité. Par exemple, le descripteur IT 2 contient un champ qui indique à l'hôte qu'il représente un connecteur externe sur le dispositif, utilisé comme une entrée de ligne analogique. De même, le descripteur MU 6 possède un champ qui indique que sa broche d'entrée 1 est connectée à la broche de sortie d'IT 1, sa broche d'entrée 2 est connectée à la broche de sortie d'IT 2 et sa broche d'entrée 3 est connectée à la broche de sortie d'IT 3.

Pour plus d'informations concernant le contenu du descripteur, se référer à la Section 4, "Descripteurs" du présent document.

Figure 3-3: Au sein de la fonction audio



Anglais	Français
Selector Unit	Unité de sélection
Feature Unit	Unité de fonction
Headphone	Casque
IT Descr	Descr. IT
SU Descr	Descr. SU
FU Descr	Descr. FU
OT Descr	Descr. OT
Analog Line	Ligne analogique
Speakers	Haut-parleurs
Analog Mic	Micro analogique
MU Descr	Descr. MU
PU Descr	Descr. PU
CS Descr	Descr. CS
CX Descr	Descr. CX
CM Descr	Descr. CM

Au sein d'une entité, la fonctionnalité est décrite plus précisément par les commandes audio. Une commande permet généralement d'accéder à une propriété audio ou d'horloge spécifique. Chaque commande possède un ensemble d'attributs qui peuvent être manipulés ou qui présentent des informations supplémentaires sur le comportement de la commande. Une commande peut avoir les attributs suivants:

- Attribut de paramètre actuel
- Triplet d'attributs de plage avec:
 - Attribut de paramètre minimum
 - Attribut de paramètre maximum
 - Attribut de résolution

Prendre pour exemple une commande de volume au sein d'une unité de fonction. En émettant les demandes "Get" appropriées, le logiciel hôte peut obtenir des valeurs pour les attributs de la commande de volume et, par exemple, les utiliser pour afficher correctement la commande sur l'écran. Le paramétrage de l'attribut actuel de la commande de volume permet au logiciel hôte de modifier le paramètre de volume de la commande de volume.

De plus, chaque entité d'une fonction audio peut avoir un attribut d'espace mémoire. Cet attribut permet en option un accès générique à l'espace mémoire interne de l'entité. Cela peut être utilisé pour mettre en œuvre la commande spécifique au fournisseur d'une entité par le biais d'un accès généralisé.

3.13.1 GRAPPE

Une grappe est un regroupement de canaux audio qui transportent des informations audio synchrones étroitement liées. Au sein de la fonction audio, il est fait abstraction complète de la forme de représentation physique réelle des données audio qui parcourent les connexions entre terminaux et unités. Chaque canal audio de la grappe est vu comme un canal logique et tous les attributs physiques du canal (largeur de bit, résolution de bit, etc.) sont non spécifiés et réputés non pertinents dans le contexte de la fonction audio du point de vue de l'interface AudioControl. Le fait qu'une broche d'entrée et une broche de sortie soient connectées ensemble dans la topologie de la fonction audio constitue une garantie (par construction) de la compatibilité des informations circulant sur la connexion avec les deux entités connectées.

La numérotation des canaux dans la grappe commence par le canal 1 et s'étend jusqu'au nombre de canaux dans la grappe. Le canal virtuel zéro est utilisé pour l'adressage d'une commande maître dans une unité (le cas échéant), influençant efficacement tous les canaux en même temps. Noter que la commande maître (le cas échéant) doit être mise en œuvre séparément des commandes par canaux. La modification du paramétrage d'une commande maître n'affecte les paramétrages d'aucune des commandes de canal individuelles. Le nombre maximal de canaux indépendants dans une grappe est limité à 255 (le canal zéro est utilisé pour référencer le canal maître).

Une grappe est caractérisée par un certain nombre d'attributs:

- un identificateur de grappe unique;
- le nombre de canaux audio dans la grappe;
- l'objet de chaque canal audio, comme la voix, la parole, etc.;
- la relation avec les autres canaux audio (ou la localisation spatiale) de chaque canal audio de la grappe;
- une indication relative à l'appartenance ou non d'un canal audio à un groupe d'ambiophonie.

Un descripteur de grappe (CD - Cluster Descriptor) est associé à chaque grappe et la décrit entièrement.

Deux types de grappes sont utilisés dans la présente spécification:

- Une grappe logique décrit les canaux audio au sein la fonction audio (la "boîte fermée") où les canaux audio sont traités comme des concepts logiques.
- Une grappe physique décrit les canaux audio physiques, par exemple lorsqu'ils traversent un connecteur.

3.13.2 TERMINAL D'ENTREE

Le terminal d'entrée (IT) est utilisé pour servir d'interface entre le "monde extérieur" de la fonction audio et d'autres unités de la fonction audio. Il constitue un réceptacle pour les informations audio entrant dans la fonction audio. Sa fonction est de représenter une source de données audio entrantes après extraction de ces données à partir du flux audio d'origine dans les canaux logiques distincts intégrés à ce flux (le processus de décodage). Les canaux logiques sont regroupés en une grappe et quittent le terminal d'entrée par une broche de sortie unique.

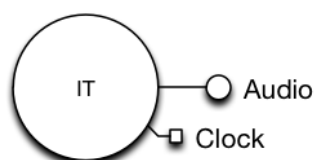
Un terminal d'entrée peut représenter des entrées de la fonction audio autres que les points d'extrémité USB OUT. Un connecteur d'entrée de ligne sur un dispositif audio est un exemple d'une telle entrée non USB. Toutefois, si le flux audio entre dans la fonction audio au moyen d'un point d'extrémité USB OUT, il existe une relation biunivoque entre l'interface AudioStreaming contenant ce point d'extrémité et son terminal d'entrée associé. Le descripteur d'interface spécifique à la classe contient un champ contenant une référence directe à ce terminal d'entrée. L'hôte a besoin d'utiliser à la fois l'interface AudioStreaming et les descripteurs de point d'extrémité ainsi que le descripteur de terminal d'entrée pour obtenir une compréhension complète des caractéristiques et capacités du terminal d'entrée. Les paramètres liés au flux sont stockés dans les descripteurs de l'interface AudioStreaming. Les paramètres liés à la commande sont stockés dans le descripteur de terminal.

Le processus de conversion de flux audio entrants, éventuellement codés, en canaux audio logiques implique toujours un type de moteur de décodage. Les types de décodages vont de schémas de décodage plutôt triviaux, comme la conversion de données MIC stéréo de 16 bits entrelacées en un canal logique gauche/droite, à des schémas très sophistiqués, comme la conversion d'un flux audio codé MPEG-2 7.1 en canaux logiques avant gauche, centre avant gauche, centre avant, centre avant droite, avant droite, arrière gauche, arrière droite et à effets basses fréquences. Le moteur de décodage est vu comme faisant partie de l'entité qui reçoit effectivement les flux de données audio codés (comme une interface USB AudioStreaming). Le type de décodage est donc déterminé par la valeur du champ **bmFormats**, situé dans le descripteur d'interface AudioStreaming spécifique à la classe. Le terminal d'entrée associé traite les canaux logiques après leur décodage.

Un terminal d'entrée possède une broche d'entrée d'horloge unique. Le signal d'horloge présent au niveau de cette broche est utilisé comme horloge d'échantillonnage pour l'ensemble du matériel sous-jacent représenté par ce terminal d'entrée. Un champ dans le descripteur de terminal d'entrée identifie de manière unique l'entité d'horloge à laquelle le terminal d'entrée est connecté.

Le symbole du terminal d'entrée est représenté sur la figure suivante:

Figure 3-4: Icône du terminal d'entrée



Anglais	Français
Clock	Horloge

3.13.3 TERMINAL DE SORTIE

Le terminal de sortie (OT) est utilisé pour servir d'interface entre les unités au sein de la fonction audio et le "monde extérieur". Il constitue une sortie pour les informations audio, sortant de la fonction audio. Sa fonction est de représenter un destinataire de données audio sortantes avant que ces données ne soient correctement regroupées, à partir des canaux logiques distincts d'origine, dans le flux audio sortant (le processus de codage). La grappe entre dans le terminal de sortie par une broche d'entrée unique.

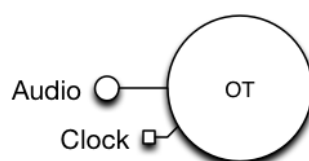
Un terminal de sortie peut représenter des sorties de la fonction audio autres que les points d'extrémité USB IN. Un haut-parleur intégré à un dispositif audio ou un connecteur de sortie de ligne constitue un exemple d'une telle sortie non USB. Toutefois, si le flux audio quitte la fonction audio au moyen d'un point d'extrémité USB IN, il existe une relation biunivoque entre l'interface AudioStreaming contenant ce point d'extrémité et son terminal de sortie associé. Le descripteur d'interface spécifique à la classe contient un champ contenant une référence directe à ce terminal de sortie. L'hôte a besoin d'utiliser à la fois l'interface AudioStreaming et les descripteurs de point d'extrémité ainsi que le descripteur de terminal de sortie pour comprendre pleinement les caractéristiques et capacités du terminal de sortie. Les paramètres liés au flux sont stockés dans les descripteurs de l'interface AudioStreaming. Les paramètres liés à la commande sont stockés dans le descripteur de terminal.

Le processus de conversion de canaux audio logiques entrants en flux audio éventuellement codés implique toujours un type de moteur de codage. Le moteur de codage est vu comme faisant partie de l'entité qui transmet effectivement les flux de données audio codés (comme une interface USB AudioStreaming). Le type de codage est donc déterminé par la valeur du champ **bmFormats**, situé dans le descripteur d'interface AudioStreaming spécifique à la classe. Le terminal de sortie associé traite les canaux logiques avant le codage.

Un terminal de sortie possède une broche d'entrée d'horloge unique. Le signal d'horloge présent au niveau de cette broche est utilisé comme horloge d'échantillonnage pour l'ensemble du matériel sous-jacent représenté par ce terminal de sortie. Un champ dans le descripteur de terminal de sortie identifie de manière unique l'entité d'horloge à laquelle le terminal de sortie est connecté.

Le symbole du terminal de sortie est représenté sur la figure suivante:

Figure 3-5: Icône du terminal de sortie



Anglais	Français
Clock	Horloge

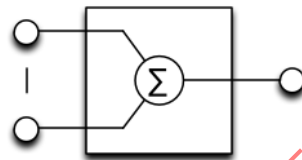
3.13.4 UNITE DE MIXAGE

L'unité de mixage (MU) transforme un certain nombre de canaux d'entrée logique en un certain nombre de canaux de sortie logiques. Les canaux d'entrée sont regroupés en une ou plusieurs grappes. Chaque grappe entre dans l'unité de mixage par une broche d'entrée. Les canaux de sortie logiques sont regroupés en une grappe et quittent l'unité de mixage par une broche de sortie unique.

Chaque canal d'entrée peut être virtuellement combiné à tous les canaux de sortie. Si n représente le nombre total de canaux d'entrée et m le nombre de canaux de sortie, alors il existe une matrice bidimensionnelle ($n \cdot m$) des commandes de mixage dans l'unité de mixage. Toutes ces commandes ne doivent pas nécessairement être mises en œuvre physiquement. Certaines commandes peuvent avoir un paramètre fixe et être non programmables. Le descripteur d'unité de mixage indique les commandes sont programmables dans le champ de topogramme binaire **bmControls**. À l'aide de ce modèle, une connexion permanente peut être mise en œuvre en indiquant la commande comme non programmable dans le topogramme binaire **bmControls** et en retournant un paramètre de commande de 0 dB lorsque cela est demandé. De même, une connexion manquante (d'inhibition) peut être mise en œuvre en indiquant la commande comme non programmable dans le topogramme binaire **bmControls** et en retournant un paramètre de commande de $-\infty$ dB. Une unité de mixage DOIT répondre à la demande "Get" appropriée pour permettre à l'hôte de récupérer les paramètres réels de chacune des commandes de mixage ($n \cdot m$).

Le symbole de l'unité de mixage peut être observé sur la figure suivante:

Figure 3-6: Icône de l'unité de mixage

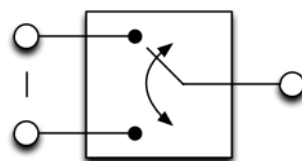


3.13.5 UNITE DE SELECTION

L'unité de sélection (SU) effectue un choix parmi n grappes, contenant chacune m canaux d'entrée logiques, et les achemine sans modification vers la grappe de sortie unique, contenant m canaux de sortie. Elle représente un sélecteur de sources de plusieurs canaux, capable de choisir entre n sources de m canaux. Elle possède n broches d'entrée et une broche de sortie unique.

Le symbole de l'unité de sélection peut être observé sur la figure suivante:

Figure 3-7: Icône de l'unité de sélection



3.13.6 UNITE DE FONCTION

L'unité de fonction (FU) est essentiellement une unité de traitement de plusieurs canaux qui permet la manipulation de base de multiples commandes audio à un seul paramètre sur les canaux logiques entrants. Pour chaque canal logique, l'unité de fonction fournit en option des commandes audio pour les fonctions suivantes:

- Inhibition
- Volume
- Commande de tonalité (Grave, Médium, Aigu)
- Egaliseur graphique
- Commande automatique de gain
- Retard

- Amplification des basses
- Intensité sonore
- Gain d'entrée
- Atténuateur de gain d'entrée
- Inverseur de phase

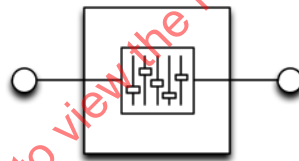
En outre, l'unité de fonctions fournit en option les commandes audio ci-dessus, mais en influençant désormais tous les canaux de la grappe en même temps. De cette façon, des commandes "maîtres" peuvent être mis en œuvre. Les commandes maîtres sont mises en cascade après les commandes de canal individuelles. Cette configuration est particulièrement utile dans les systèmes multicanaux, où les commandes de canal individuelles peuvent être utilisées pour l'équilibrage de canal et les commandes maîtres peuvent être utilisées pour les paramètres généraux.

Les canaux logiques de la grappe sont numérotés à partir de un jusqu'au nombre total de canaux dans la grappe. Le canal "maître" porte le numéro de canal zéro et est toujours virtuellement présent.

Le descripteur d'unité de fonction indique les commandes présentes pour chaque canal de l'unité de fonction et pour le canal "maître". Tous les canaux logiques d'une unité de fonction sont entièrement indépendants. Il n'existe pas de couplage croisé entre les canaux au sein de l'unité de fonction. Il y a autant de canaux de sortie logiques que de canaux d'entrée. Ceux-ci sont regroupés en une grappe qui entre dans l'unité de fonction par le biais d'une broche d'entrée unique et quitte l'unité par une broche de sortie unique.

Le symbole de l'unité de fonction est représenté sur la figure suivante:

Figure 3-8: Icône de l'unité de fonction



3.13.7 UNITE DE CONVERSION DU DEBIT D'ECHANTILLONNAGE

L'unité de conversion du débit d'échantillonnage (RU) est incluse ici comme une manière facultative d'indiquer exactement où la conversion du débit d'échantillonnage de la fonction audio a lieu. Dans de nombreux cas, il n'est pas nécessaire d'indiquer ce point et toute unité de conversion du débit d'échantillonnage peut être omise de la topologie sans affecter matériellement les informations présentées à l'hôte. La raison principale de l'inclusion de l'unité de conversion du débit d'échantillonnage est d'indiquer avec précision toute latence subie par le processus de conversion du débit d'échantillonnage.

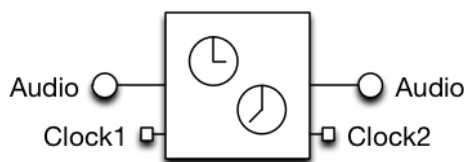
L'unité de conversion du débit d'échantillonnage fournit une fonction de pont entre différents domaines d'horloge au sein de la fonction audio. Le convertisseur de débit d'échantillonnage ne fournit aucune commande audio. Il prélève l'audio de tous les canaux logiques de la grappe d'entrée unique appartenant à un domaine d'horloge donné et les convertit en mêmes canaux logiques dans la grappe de sortie unique, mais appartenant désormais à un autre domaine d'horloge.

Il y a autant de canaux de sortie logiques que de canaux d'entrée. Ceux-ci sont regroupés en une grappe qui entre dans l'unité de conversion du débit d'échantillonnage par le biais d'une broche d'entrée unique et quitte l'unité par une broche de sortie unique.

Une unité de conversion du débit d'échantillonnage possède deux broches d'entrée d'horloge. Une broche d'entrée d'horloge est associée à la broche d'entrée unique de l'unité de conversion du débit d'échantillonnage. L'autre broche d'entrée d'horloge est associée à la broche de sortie unique de l'unité de conversion du débit d'échantillonnage. Les signaux d'horloge présents au niveau de ces deux broches d'entrée d'horloge identifient les deux domaines d'horloge entre lesquels l'unité de conversion du débit d'échantillonnage effectue la conversion. Noter qu'il est admis d'avoir les deux broches d'entrée d'horloge connectées à des signaux d'horloge appartenant au même domaine d'horloge.

Le symbole de l'unité de conversion du débit d'échantillonnage est représenté sur la figure suivante:

Figure 3-9: Icône de l'unité de conversion du débit d'échantillonnage



Anglais	Français
Clock	Horloge

3.13.8 UNITE D'EFFET

L'unité d'effet (EU) est une unité de traitement de plusieurs canaux qui permet la manipulation avancée d'une commande audio à plusieurs paramètres sur les canaux logiques entrants, canal par canal. Pour chaque canal logique, l'unité d'effet fournit l'une des commandes audio suivantes:

- Section d'égaliseur paramétrique
- Réverbération
- Retard de modulation
- Compresseur de plage dynamique

En outre, l'unité d'effet fournit en option l'une des commandes audio ci-dessus, mais en influençant désormais tous les canaux de la grappe en même temps. De cette façon, une commande "maître" peut être mise en œuvre. La commande maître est mise en cascade après les commandes de canal individuelles. Cette configuration est particulièrement utile dans les systèmes multicanaux, où les commandes de canal individuelles peuvent être utilisées pour l'équilibrage de canal et la commande maître peut être utilisée pour les paramètres généraux.

Les canaux logiques de la grappe sont numérotés à partir de un jusqu'au nombre total de canaux dans la grappe. Le canal "maître" porte le numéro de canal zéro et est toujours virtuellement présent.

Le descripteur d'unité d'effet indique les commandes présentes pour chaque canal de l'unité d'effet et pour le canal "maître". Tous les canaux logiques d'une unité d'effet sont entièrement indépendants. Il n'existe pas de couplage croisé entre les canaux au sein de l'unité d'effet. Il y a autant de canaux de sortie logiques que de canaux d'entrée. Ceux-ci sont regroupés en une grappe qui entre dans l'unité d'effet par le biais d'une broche d'entrée unique et quitte l'unité par une broche de sortie unique.

Si l'unité d'effet est un membre explicite d'un domaine d'alimentation, le basculement du domaine d'alimentation vers un état d'alimentation autre que D0 doit rendre l'unité non fonctionnelle et sa sortie n'est pas définie. Une fonctionnalité de contournement explicite peut être exigée pour préserver l'intégrité du chemin de signal.

3.13.8.1 UNITE D'EFFET DE SECTION D'EGALISEUR PARAMETRIQUE

L'unité d'effet de section d'égaliseur paramétrique (PEQS - Parametric Equalizer Section) est utilisée pour manipuler et égaliser les caractéristiques de fréquence des informations audio d'origine autour d'une fréquence médiane donnée. Afin de construire un égaliseur paramétrique, il peut être nécessaire de mettre en cascade un certain nombre de ces unités d'effet PEQS pour obtenir la fonctionnalité souhaitée. Les paramètres qui peuvent être manipulés pour obtenir l'effet d'égalisation souhaité sont les suivants:

- Fréquence médiane: fréquence autour de laquelle le spectre audio est manipulé. Exprimée en Hz.
- Facteur Q: mesure de la plage des fréquences autour de la fréquence médiane qui sont influencées. Exprimé sous forme de rapport.
- Gain: quantité de gain ou d'atténuation à la fréquence médiane. Exprimé en dB.

L'algorithme permettant de produire l'effet d'égalisation souhaité peut être manipulé canal par canal. Le concept de canal maître permet d'égaliser tous les canaux simultanément.

Le symbole de l'unité de traitement PEQS peut être observé sur la figure suivante:

Figure 3-10: Icône de l'unité d'effet PEQS



3.13.8.2 UNITE D'EFFET DE REVERBERATION

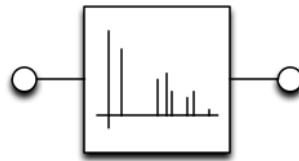
L'unité d'effet de réverbération est utilisée pour ajouter des effets d'acoustique de salle à l'information audio d'origine. Ces effets peuvent aller des effets de réverbération de petites salles à la simulation de la réverbération de grandes salles de concert. Un certain nombre de paramètres peuvent être manipulés pour obtenir les effets de réverbération souhaités.

- Type de réverbération: Room1, Room2, Room3, Hall1, Hall2, Plaque, Retard et Retard de répétition.
- Niveau de réverbération: définit la quantité de son réverbérant par rapport au son d'origine. Exprimé sous forme de rapport.
- Temps de réverbération: définit la durée pendant laquelle la réverbération continue. Exprimé en s.
- Rétroaction de retard de réverbération: utilisée avec les types de réverbération Retard et Retard de répétition. Définit la façon dont le retard se répète. Exprimée sous forme de rapport.
- Pré-retard de réverbération: définit le temps de retard entre le son d'origine et l'écho de réverbération initial. Exprimé en ms.
- Densité de réverbération: définit la densité des échos de réverbération.
- Affaiblissement de réverbération haute fréquence: définit la fréquence de coupure d'un filtre passe-bas sur les échos. Exprimé en Hz.

La manière dont un certain effet de réverbération est obtenu relève entièrement du concepteur. La présente spécification n'a pas pour objet de définir précisément tous les paramètres qui influencent l'expérience de réverbération (par exemple, dans un système multicanal, il est possible de créer des impressions de réverbération hautement similaires, en utilisant différents algorithmes et réglages de paramètres sur l'ensemble des canaux).

Le symbole de l'unité d'effet de réverbération peut être observé sur la figure suivante:

Figure 3-11: Icône de l'unité d'effet de réverbération



3.13.8.3 UNITE D'EFFET DE RETARD DE MODULATION

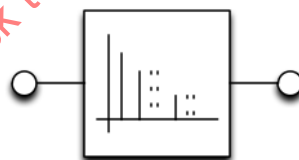
L'unité d'effet de retard de modulation est utilisée pour ajouter des effets de modulation (comme un chœur) aux informations audio d'origine. Un certain nombre de paramètres peuvent être manipulés pour obtenir les effets de modulation souhaités.

- Equilibre de retard de modulation: contrôle le rapport du son d'origine en fonction de celui du son avec effet. Exprimé sous forme de rapport.
- Taux de retard de modulation: définit la vitesse (fréquence) du modulateur. Exprimé en Hz.
- Profondeur de retard de modulation: définit la profondeur à laquelle le son est modulé. Exprimée en ms.
- Temps de retard de modulation: définit le retard qui est ajouté au son modulé avant de l'ajouter au son d'origine. Exprimé en ms.
- Niveau de rétroaction de retard de modulation: contrôle la quantité de son modulé qui est réacheminée vers l'entrée de l'unité de modulation. Exprimé sous forme de rapport.

La manière dont un certain effet de modulation est obtenu relève entièrement du concepteur.

Le symbole de l'unité d'effet de retard de modulation peut être observé sur la figure suivante:

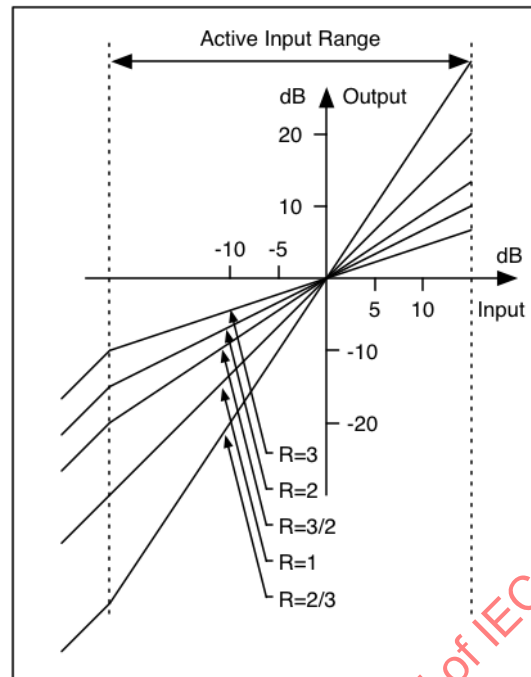
Figure 3-12: Icône de l'unité d'effet de retard de modulation



3.13.8.4 UNITE D'EFFET DE COMPRESSEUR DE PLAGE DYNAMIQUE

L'unité d'effet de compresseur de plage dynamique est utilisée pour limiter de manière intelligente la plage dynamique des informations audio d'origine. Un certain nombre de paramètres peuvent être manipulés pour influencer la compression souhaitée.

Figure 3-13: Caractéristique de transfert du compresseur de plage dynamique



Anglais	Français
Active Input Range	Plage d'entrée active
Output	Sortie
Input	Entrée

- Taux de compression R: détermine la pente de la caractéristique de transfert entrée-sortie statique dans la plage d'entrée active du compresseur. La compression est définie par le taux de compression R, qui est l'inverse du dérivé de la puissance de sortie P_o en fonction de la puissance d'entrée P_i , avec P_o et P_i exprimées en dB.

$$R^{-1} = \frac{\partial \text{Log}(\frac{P_o}{P_R})}{\partial \text{Log}(\frac{P_i}{P_R})}$$

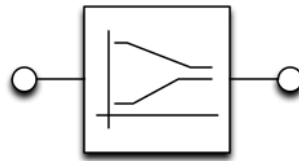
P_R est le niveau de référence et est rendu égal au niveau dit de ligne. Tous les niveaux sont exprimés par rapport au niveau de ligne (0 dB), qui se situe généralement 15 à 20 dB au-dessous du niveau maximal. La compression est obtenue lorsque $R > 1$, $R = 1$ n'affectant pas le signal et $R < 1$ donnant lieu à une expansion.

- Amplitude maximale: limite supérieure de la plage d'entrée active, par rapport au niveau de ligne (0 dB). Exprimée en dB.
- Niveau de seuil: limite inférieure du niveau d'entrée actif, par rapport au niveau de ligne (0 dB).
- Temps d'attaque: détermine la réponse du compresseur en fonction du temps à un palier du niveau d'entrée. Exprimé en ms.
- Temps de relâchement: se rapporte au temps de récupération du gain du compresseur après un passage fort. Exprimé en ms.
- Gain "Make-up": définit pour compenser la perte de gain dans le compresseur. Exprimé en dB.

La manière dont une certaine compression de plage dynamique est obtenue relève entièrement du concepteur.

Le symbole de l'unité d'effet de compresseur de plage dynamique peut être observé sur la figure suivante:

Figure 3-14: Icône de l'unité d'effet de compresseur de plage dynamique



3.13.9 UNITE DE TRAITEMENT

L'unité de traitement (PU) représente un bloc fonctionnel au sein de la fonction audio qui transforme un certain nombre de canaux d'entrée logiques, regroupés en une ou plusieurs grappes, en un certain nombre de canaux de sortie logiques, regroupés en une seule grappe. Par conséquent, l'unité de traitement peut posséder plusieurs broches d'entrée et possède une broche de sortie unique. La présente spécification définit plusieurs transformées (algorithmes) normalisées, réputées nécessaires pour la prise en charge de fonctionnalités audio supplémentaires; ces transformées ne sont pas couvertes par les autres types d'unités, mais elles sont assez répandues pour être incluses dans la présente spécification, de sorte qu'un pilote générique puisse en assurer le contrôle.

S'il est nécessaire d'être capable de contourner la fonctionnalité intégrée à l'unité de traitement, alors une topologie de contournement explicite utilisant une unité de sélection doit être mise en œuvre.

Si l'unité de traitement est un membre explicite d'un domaine d'alimentation, le basculement du domaine d'alimentation vers un état d'alimentation autre que D0 doit rendre l'unité non fonctionnelle et sa sortie n'est pas définie. Une fonctionnalité de contournement explicite peut être exigée pour préserver l'intégrité du chemin de signal.

3.13.9.1 UNITE DE TRAITEMENT DE MIXAGE ELEVEUR/REDUCTEUR

L'unité de traitement de mixage élévateur/réducteur fournit des mécanismes permettant de dériver m canaux audio de sortie à partir de n canaux audio d'entrée. Les algorithmes et transformées appliqués à cette fin ne sont pas définis par la présente spécification et peuvent être propriétaires. Les canaux d'entrée sont regroupés en une grappe de canaux d'entrée qui entre dans l'unité de traitement par une broche d'entrée unique. De même, tous les canaux de sortie sont regroupés en une grappe de canaux de sortie, quittant l'unité de traitement par une broche de sortie unique.

L'unité de traitement de mixage élévateur/réducteur peut prendre en charge plusieurs modes de fonctionnement. Les canaux audio d'entrée disponibles sont dictés par l'entité à laquelle l'unité de traitement de mixage élévateur/réducteur est connectée. Le descripteur d'unité de traitement de mixage élévateur/réducteur indique les modes de mixage élévateur/réducteur pris en charge par l'unité au moyen de sa matrice **waClusterDescrID()**. Chaque élément de la matrice **waClusterDescrID()** indique quels canaux de sortie de la grappe de sortie sont effectivement utilisés dans un mode particulier. Les canaux de sortie non utilisés de la grappe de sortie doivent produire une sortie muette. Le choix du mode est mis en œuvre à l'aide de la demande de commande de choix de mode "Get/Set Mode Select Control".

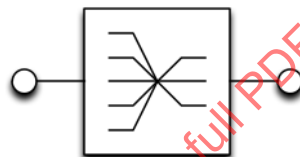
Prendre pour exemple le cas d'une unité de traitement de mixage élévateur/réducteur connectée à un terminal d'entrée, produisant un audio décodé Dolby™ AC-3 5.1. La grappe d'entrée de l'unité de traitement de mixage élévateur/réducteur contient donc des canaux logiques de gauche, droite, centre, surround gauche, surround droite et LFE (Low Frequency Effects, effets basses fréquences).

Prendre pour hypothèse que le matériel de la fonction audio se limite à la reproduction audio double canal uniquement. L'unité de traitement de mixage élévateur/réducteur peut ensuite utiliser des algorithmes (sophistiqués) pour soumettre les informations audio spatiales disponibles à un mixage réducteur en deux canaux ("enrichis") de manière à pouvoir obtenir une expérience maximale des effets spatiaux, en utilisant seulement deux canaux. L'utilisation de l'algorithme de mixage réducteur approprié en fonction de la nature physique du terminal de sortie vers lequel l'unité de traitement de mixage élévateur/réducteur est acheminée relève de la mise en œuvre. Par exemple, un algorithme de mixage réducteur différent est nécessaire selon si le flux stéréo "enrichi" est envoyé à une paire de haut-parleurs ou à un casque. Toutefois, cette connaissance réside d'ores et déjà dans la fonction audio et l'intervention de l'hôte n'est pas nécessaire pour décider quel algorithme de mixage réducteur utiliser.

En tant que deuxième exemple intéressant, prendre pour hypothèse que le matériel est capable de desservir huit canaux audio distincts (par exemple, un système MPEG-2 7.1 intégral). L'unité de traitement de mixage élévateur/réducteur peut maintenant utiliser certaines techniques afin de dériver du contenu pertinent pour les canaux audio supplémentaires (centre gauche, centre droite) présents dans la grappe de sortie et absents dans la grappe de canaux d'entrée (AC-3 5.1). Il s'agit d'un exemple type de mixage élévateur.

Le symbole de l'unité de traitement de mixage élévateur/réducteur est représenté sur la figure suivante:

Figure 3-15: Icône de l'unité de traitement de mixage élévateur/réducteur

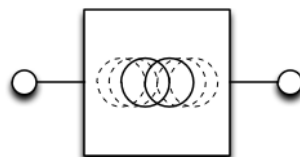


3.13.9.2 UNITE DE TRAITEMENT D'AMPLIFICATION STEREO

L'unité de traitement d'amplification stéréo concerne uniquement les canaux gauche et droite. Elle procède au traitement d'une bande sonore stéréo (deux canaux) existante en vue d'étendre le champ acoustique et de le faire apparaître en provenance de l'extérieur des emplacements de haut-parleurs avant gauche/droite. Des effets stéréo étendus peuvent être obtenus à l'aide de différentes méthodes. Les algorithmes et transformées appliqués à cette fin ne sont pas définis par la présente spécification et peuvent être propriétaires. La largeur perçue du champ acoustique peut être contrôlée au moyen de la demande "Get/Set Width Control".

Le symbole de l'unité d'amplification stéréo est représenté sur la figure suivante:

Figure 3-16: Icône de l'unité de traitement d'amplification stéréo



3.13.9.3 UNITE DE TRAITEMENT MULTIFONCTION

L'unité de traitement multifonction regroupe différents blocs algorithmiques liés, fournissant ensemble certaines fonctionnalités. La présente spécification définit plusieurs algorithmes réputés utiles et assez répandus pour être inclus dans la présente spécification.

Les algorithmes suivants sont actuellement pris en charge:

- Algorithme non défini
- Algorithme de formation de faisceau
- Algorithme d'annulation d'écho acoustique
- Algorithme d'annulation active du bruit
- Algorithme de séparation de source aveugle
- Suppression/réduction du bruit

Le descripteur d'unité de traitement multifonction contient un topogramme binaire indiquant les algorithmes pris en charge par l'unité. La mise en œuvre exacte de ces algorithmes et la manière dont ils peuvent interagir dépendent de la mise en œuvre.

Note 1: S'il est nécessaire d'exposer à l'hôte la manière dont les algorithmes sont interconnectés, un concepteur peut choisir de modéliser l'ensemble des algorithmes en utilisant plusieurs unités de traitement multifonctions, chacune ne contenant qu'un seul algorithme ou bien un sous-ensemble d'algorithmes, et en les connectant explicitement.

Note 2: La plupart des algorithmes mentionnés ci-dessus impliquent une forme de traitement de signal qui ne peut pas être admise comme étant linéaire et invariante dans le temps.

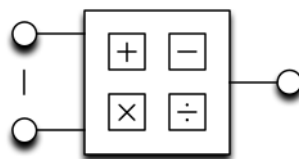
En fonctionnement normal, l'unité de traitement multifonction transforme un certain nombre de canaux d'entrée logiques entrant dans l'unité par une ou plusieurs broches d'entrée en un certain nombre de canaux de sortie logiques quittant l'unité par une broche de sortie unique.

S'il est nécessaire d'être capable de contourner la fonctionnalité intégrée à l'unité de traitement multifonction, alors une topologie de contournement explicite utilisant une unité de sélection doit être mise en œuvre.

Il est fortement recommandé de mettre en œuvre la fonctionnalité de contournement, de sorte qu'un pilote audio générique qui ne comprend pas quelle fonctionnalité est mise en œuvre dans l'unité de traitement multifonction soit capable de la supprimer du chemin de signal.

Le symbole de l'unité de traitement multifonction est représenté sur la figure suivante:

Figure 3-17 Icône de l'unité de traitement multifonction



3.13.10 UNITE D'EXTENSION

L'unité d'extension (XU) est la méthode fournie par la présente spécification pour ajouter facilement des blocs de construction spécifiques au fournisseur à la spécification. L'unité d'extension fournit un ou plusieurs canaux d'entrée logiques, regroupés en une ou plusieurs grappes, et les transforme en plusieurs canaux de sortie logiques, regroupés en une seule grappe. Par conséquent, l'unité d'extension peut posséder plusieurs broches d'entrée et possède une broche de sortie unique.

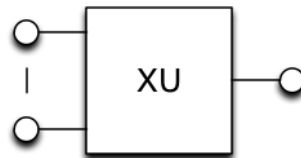
S'il est nécessaire d'être capable de contourner la fonctionnalité intégrée à l'unité d'extension, alors une topologie de contournement explicite utilisant une unité de sélection doit être mise en œuvre.

Il est fortement recommandé de mettre en œuvre la fonctionnalité de contournement, de sorte qu'un pilote audio générique qui ne comprend pas quelle fonctionnalité est mise en œuvre dans l'unité d'extension soit capable de la supprimer du chemin de signal.

Si l'unité d'extension est un membre explicite d'un domaine d'alimentation, le basculement du domaine d'alimentation vers un état d'alimentation autre que D0 doit rendre l'unité non fonctionnelle et sa sortie n'est pas définie. Une fonctionnalité de contournement explicite peut être exigée pour préserver l'intégrité du chemin de signal.

Le symbole de l'unité d'extension peut être observé sur la figure suivante:

Figure 3-18: Icône de l'unité d'extension



3.13.11 ENTITES D'HORLOGE

Les entités d'horloge sont particulières, dans le sens où elles ne manipulent pas directement les flux audio logiques. Au lieu de cela, elles fournissent la fonctionnalité nécessaire pour manipuler les signaux d'horloge d'échantillonnage et le routage d'horloge des différents terminaux d'entrée et de sortie au sein de la fonction audio. Un terminal au sein de la fonction audio ne peut être connecté qu'à une seule entité d'horloge. Le signal d'horloge présent au niveau de la broche d'entrée d'horloge d'un terminal détermine la fréquence d'échantillonnage à laquelle le matériel sous-jacent fonctionne.

3.13.11.1 SOURCE D'HORLOGE

Une entité source d'horloge fournit un signal d'horloge d'échantillonnage indépendant sur sa broche de sortie d'horloge unique. L'entité source d'horloge sert d'horloge maître pour un domaine d'horloge. Plusieurs différentes autres fréquences d'échantillonnage synchrones peuvent être dérivées en connectant plusieurs entités de multiplicateurs d'horloge à la même entité source d'horloge. En manipulant la commande de fréquence d'échantillonnage au sein de l'entité source d'horloge, toutes les fréquences d'échantillonnage du domaine d'horloge sont influencées.

Chaque horloge d'échantillonnage maître indépendante au sein de la fonction audio doit être représentée par une entité source d'horloge distincte. Même si l'horloge est générée "au sein d'un terminal", il est nécessaire de représenter cette horloge par une entité source d'horloge. Par exemple, une horloge d'échantillonnage peut être récupérée en fonction de la quantité d'échantillons audio entrant dans la fonction audio sur un point d'extrémité adaptatif USB OUT. En variante, une horloge d'échantillonnage peut être dérivée à partir du signal S/PDIF entrant dans la fonction audio sur un connecteur externe.

Note: Dans le cas d'un point d'extrémité de données isochrone adaptatif ne prenant en charge qu'un nombre discret de fréquences d'échantillonnage, le point d'extrémité doit au moins tolérer une inexactitude de ± 1000 PPM sur les valeurs de commande de fréquence d'échantillonnage déclarées, afin de tenir compte des inexactitudes d'horloge d'échantillonnage.

Le descripteur d'entité source d'horloge contient un champ qui indique l'origine du signal d'horloge réel, représenté par l'entité. De plus, étant donné que les terminaux d'entrée et de sortie ne possèdent qu'une broche d'entrée d'horloge, un signal d'horloge ne peut jamais être généré directement à partir d'un terminal.

La sortie d'une entité source d'horloge n'a pas à être valide à tout moment. Par exemple, si une entité source d'horloge représente une entrée d'horloge d'échantillonnage externe sur la fonction audio, la sortie de cette source d'horloge peut ne pas être valide lorsqu'il n'y a aucune connexion à l'entrée d'horloge externe. La source d'horloge peut être interrogée quant à la validité de son signal de sortie à tout moment.

Le symbole de l'entité source d'horloge peut être observé sur la figure suivante:

Figure 3-19: Icône de la source d'horloge



3.13.11.2 SELECTEUR D'HORLOGE

Une entité de sélecteur d'horloge fournit la fonctionnalité permettant d'effectuer un choix parmi les différents signaux d'horloge d'échantillonnage disponibles. Elle peut avoir n broches d'entrée d'horloge à partir desquelles l'un des signaux est acheminé vers la broche de sortie d'horloge unique.

Le basculement entre les entrées d'horloge peut être commandé par l'hôte (le sélecteur d'horloge est programmable à travers la demande "Set" appropriée) ou la fonction audio peut faire basculer les entrées d'horloge en raison d'un événement externe. Un sélecteur d'horloge peut prendre en charge les deux méthodes de commande. Le sélecteur d'horloge peut informer l'hôte de la modification en générant l'interruption appropriée.

Le symbole de l'entité de sélecteur d'horloge peut être observé sur la figure suivante:

Figure 3-20: Icône du sélecteur d'horloge



3.13.11.3 MULTIPLICATEUR D'HORLOGE

Une entité de multiplicateur d'horloge fournit la fonctionnalité permettant de dériver une nouvelle fréquence de signal d'horloge d'échantillonnage à partir du signal d'horloge d'échantillonnage, présent au niveau de sa broche d'entrée d'horloge unique. L'algorithme utilisé pour dériver le nouveau signal d'horloge n'est pas défini par la présente spécification. Toutefois, il existe une exigence selon laquelle le signal d'horloge de sortie doit être synchrone au signal d'horloge d'entrée, de sorte que les deux horloges appartiennent au même domaine d'horloge. Une entité de multiplicateur d'horloge contient un multiplicateur suivi d'un diviseur. Le facteur de multiplication P et le facteur de division Q peuvent être programmables et situés dans la plage $[1, 2^{16}-1]$. La fréquence d'échantillonnage qui en résulte est obtenue en multipliant la fréquence du signal d'entrée par P/Q.

Le symbole de l'entité de multiplicateur d'horloge peut être observé sur la figure suivante:

Figure 3-21: Icône du multiplicateur d'horloge



3.14 MODELE OPERATIONNEL

Un dispositif peut prendre en charge plusieurs configurations. Dans chaque configuration, il peut y avoir plusieurs interfaces, chacune présentant éventuellement des paramètres alternatifs. Ces interfaces peuvent concerner différentes fonctions cohabitant dans le même dispositif composite. Plusieurs fonctions audio indépendantes peuvent aussi exister dans le même dispositif. Les interfaces appartenant à la même fonction audio sont regroupées en une association d'interfaces audio. Si le dispositif contient plusieurs fonctions audio indépendantes, il doit y avoir plusieurs associations d'interfaces audio, fournissant chacune un accès total à sa fonction audio associée.

Prendre pour exemple de dispositif composite un moniteur PC équipé d'un système de haut-parleurs stéréo intégré. Un tel dispositif peut être configuré de manière à avoir une interface pour la configuration et la commande de la partie moniteur du dispositif (classe HID), tandis qu'un ensemble de deux autres interfaces concerne ses aspects audio. L'une d'elles, l'interface AudioControl, est utilisée pour contrôler le fonctionnement interne de la fonction (commande de volume, etc.) tandis que l'autre, l'interface AudioStreaming, gère le trafic des données envoyées au sous-système audio du moniteur.

L'interface AudioStreaming peut être configurée pour fonctionner en mode mono (paramètre alternatif x), dans lequel seul un flux de données de canal unique est envoyé à la fonction Audio. Le terminal d'entrée de réception peut dupliquer ce flux audio en deux canaux logiques, et ceux-ci peuvent ensuite être reproduits sur les deux haut-parleurs. Du point de vue d'une interface, une telle configuration exige un point d'extrémité isochrone dans l'interface AudioStreaming pour recevoir le flux de données audio mono, en plus du point d'extrémité de commande obligatoire et du point d'extrémité d'interruption facultatif dans l'interface AudioControl.

Le même système peut être utilisé pour la lecture audio stéréo. Dans ce cas, l'interface AudioStreaming stéréo est sélectionnée (paramètre alternatif y). Cette interface se compose également d'un point d'extrémité isochrone unique, qui reçoit à présent un flux de données qui entrelace des échantillons de canaux avant gauche et avant droit. Le terminal d'entrée de réception divise à présent le flux en un canal logique avant gauche et un canal logique avant droit. Le paramètre alternatif de l'interface AudioControl reste inchangé.

Si l'interface AudioStreaming ci-dessus était un destinataire asynchrone, un point d'extrémité de rétroaction isochrone supplémentaire serait également nécessaire.

Comme indiqué précédemment, la fonctionnalité audio se situe au niveau de l'interface dans la hiérarchie des classes de dispositifs. Les sections suivantes décrivent l'association d'interfaces audio, comprenant une interface AudioControl unique et des interfaces AudioStreaming facultatives, ainsi que leurs points d'extrémité associés, utilisés pour le contrôle de la fonction audio et pour le transfert de flux de données audio.

3.14.1 INTERFACE AUDIOCONTROL

Afin de contrôler le comportement fonctionnel d'une fonction audio en particulier, l'hôte peut manipuler les entités d'horloge, les unités et les terminaux au sein de la fonction audio. Pour rendre ces objets accessibles, la fonction audio doit exposer une interface AudioControl unique. Cette interface peut contenir les points d'extrémité suivants:

- Un point d'extrémité de commande permettant de manipuler les paramètres d'entité d'horloge, d'unité et de terminal et de récupérer l'état de la fonction audio. Ce point d'extrémité est obligatoire, et le point d'extrémité par défaut 0 est utilisé à cette fin.
- Un point d'extrémité d'interruption. Ce point d'extrémité est facultatif, mais il doit être mis en œuvre si l'une des commandes au sein du dispositif a besoin d'interrompre l'hôte pour l'informer d'un changement de comportement de la fonction audio.

L'interface AudioControl est le point d'entrée unique permettant d'accéder aux composantes internes de la fonction audio. Toutes les demandes concernées par la manipulation de certaines commandes audio au sein des entités d'horloge, unités ou terminaux de la fonction audio doivent être dirigées vers l'interface AudioControl de la fonction audio. De même, tous les descripteurs liés aux composantes internes de la fonction audio font partie du descripteur d'interface AudioControl spécifique à la classe.

L'interface AudioControl d'une fonction audio ne peut prendre en charge qu'un seul paramètre alternatif (paramètre alternatif 0).

3.14.1.1 POINT D'EXTREMITÉ DE COMMANDE

La classe d'interface audio utilise le point d'extrémité 0 (le canal par défaut) comme méthode normalisée de commande de la fonction audio, à l'aide de demandes spécifiques à la classe. Ces demandes sont toujours adressées à l'une des entités d'horloge, l'une des unités ou l'un des terminaux constituant la fonction audio. Le format et le contenu de ces demandes sont décrits plus loin dans le présent document.

3.14.1.2 POINT D'EXTREMITÉ D'INTERRUPTION

Une interface USB AudioControl peut prendre en charge un point d'extrémité d'interruption facultatif afin d'informer l'hôte des modifications dynamiques qui apparaissent sur les différentes entités adressables (entités d'horloge, terminaux, unités, interfaces et points d'extrémité) au sein de la fonction audio. Le point d'extrémité d'interruption est utilisé par l'ensemble de l'association d'interfaces audio pour transmettre des informations de modification à l'hôte. Il est vu comme faisant partie de l'interface AudioControl, car il s'agit de l'interface d'ancrage de l'ensemble.

3.14.2 INTERFACE AUDIOSTREAMING

Les interfaces AudioStreaming sont utilisées pour l'échange de flux de données audio numériques entre l'hôte et la fonction audio. Elles sont facultatives. Une fonction audio peut n'avoir aucune ou avoir plusieurs interfaces AudioStreaming associées, chacune transportant éventuellement des données de nature et de format différents. Chaque interface AudioStreaming peut avoir au plus un point d'extrémité de données isochrone. Cette construction garantit une relation biunivoque entre l'interface AudioStreaming et le flux de données audio unique, lié au point d'extrémité. Dans certains cas, le point d'extrémité de données isochrone est accompagné d'un point d'extrémité de rétroaction explicite isochrone associé à des fins de synchronisation. Le point d'extrémité de données isochrone et son point d'extrémité de rétroaction associé doivent suivre le schéma de numérotation des points d'extrémité défini dans la *Spécification USB*.

Une interface AudioStreaming peut avoir des paramètres alternatifs qui peuvent être utilisés pour modifier certaines caractéristiques de l'interface et du point d'extrémité sous-jacent. Une utilisation type des paramètres alternatifs consiste à fournir un moyen de modifier la taille de la sous-trame et/ou le nombre de canaux d'une interface AudioStreaming active. Chaque fois qu'une interface AudioStreaming exige un point d'extrémité de données isochrone, elle doit au moins fournir le paramètre alternatif par défaut (paramètre alternatif 0) avec des exigences de bande passante nulle (aucun point d'extrémité de données isochrone défini) et un paramètre alternatif supplémentaire qui contient le point d'extrémité de données isochrone réel.

L'interface AudioStreaming est essentiellement utilisée pour fournir un point d'accès au logiciel hôte (pilotes) afin de manipuler le comportement de l'interface physique qu'elle représente. Par conséquent, même les connexions externes à la fonction audio (interface S/PDIF, entrée analogique, etc.) peuvent être représentées par une interface AudioStreaming, de sorte que le logiciel hôte puisse contrôler certains aspects de ces connexions. Ce type d'interface AudioStreaming n'a pas de points d'extrémité USB associés. Le flux de données audio associé n'utilise pas l'USB comme moyen de transport.

Pour chaque point d'extrémité OUT ou IN isochrone défini dans l'une des interfaces AudioStreaming, il doit exister un terminal d'entrée ou de sortie correspondant défini dans la fonction audio. Pour que l'hôte comprenne pleinement la nature et le comportement de la connexion, il a besoin de prendre en compte les descripteurs liés à l'interface et au point d'extrémité ainsi que le descripteur lié au terminal.

3.14.2.1 POINT D'EXTREMITE DE FLUX DE DONNEES AUDIO ISOCHRONE

En général, les flux de données gérés par un point d'extrémité de données audio isochrone ne sont pas nécessairement mis en correspondance directe avec les canaux logiques existant au sein la fonction audio. Prendre pour exemple le cas de plusieurs canaux audio logiques compressés en un flux de données (AC-3, WMA...) unique. Le format d'un tel flux de données peut être entièrement différent du format natif des canaux logiques (par exemple, audio AC-3 5.1 640 Kbits/s par opposition à 6 canaux audio 44,1 kHz de 16 bits). Par conséquent, pour décrire correctement le transfert de données au niveau du point d'extrémité, la notion de canal logique est remplacée par la notion de flux de données audio. Il incombe à l'interface AudioStreaming contenant le point d'extrémité OUT d'effectuer la conversion entre le flux de données audio et les canaux logiques intégrés avant de remettre les données au terminal d'entrée. Dans de nombreux cas, ce processus de conversion implique une forme de décodage. De même, l'interface AudioStreaming contenant le point d'extrémité IN doit convertir les canaux logiques en provenance du terminal de sortie en un flux de données audio, souvent en utilisant une forme de codage. Si le processus de décodage ou de codage expose les commandes qui influencent le codage ou le décodage, ces commandes peuvent être accessibles par le biais de l'interface AudioStreaming.

Les demandes relatives au contrôle des propriétés existant au sein d'une fonction audio, telles que le volume ou l'inhibition, ne peuvent pas être envoyées au point d'extrémité d'une interface AudioStreaming. Une interface AudioStreaming concerne les flux de données audio et n'a pas connaissance du nombre de canaux logiques qu'elle dessert. Au lieu de cela, ces demandes doivent être adressées aux unités ou terminaux de la fonction audio appropriée par l'intermédiaire de l'interface AudioControl.

Comme mentionné précédemment, une interface AudioStreaming peut n'avoir aucun ou avoir un seul point d'extrémité de données audio isochrone. Si plusieurs canaux audio synchrones doivent être communiqués entre l'hôte et la fonction audio, ils doivent être regroupés en une grappe physique en entrelaçant les données audio individuelles, et le résultat peut être dirigé vers le point d'extrémité unique.

Si une fonction audio a besoin de plus d'une grappe pour fonctionner, chaque grappe est dirigée vers le point d'extrémité d'une interface AudioStreaming distincte, appartenant à la même association d'interfaces audio (desservant toutes la même fonction audio).

3.14.2.2 POINT D'EXTREMITE DE RETROACTION ISOCHRONÉ

Pour les points d'extrémité sources audio adaptatifs et les points d'extrémité destinataires audio asynchrones, un mécanisme de synchronisation explicite est nécessaire afin de maintenir la synchronisation pendant les transferts. Pour plus d'informations concernant la synchronisation à différentes vitesses de point d'extrémité, voir la *Spécification USB* applicable.

3.14.2.3 FORMAT DE DONNEES AUDIO

Le format utilisé pour transporter des données audio sur l'USB est entièrement déterminé par les bits définis dans le champ **bmFormats** du descripteur d'interface spécifique à la classe. Des champs supplémentaires dans ce descripteur décrivent plus précisément le format. Pour plus d'informations sur les types de format définis et les formats de données associés, voir le document séparé *Formats de données audio USB*, qui est vu comme faisant partie de la présente spécification. Les protocoles spécifiques au fournisseur doivent être entièrement documentés par le fabricant.

3.14.3 MODELE D'HORLOGE

Les entités d'horloge fournissent un moyen de décrire avec précision l'utilisation et la distribution des signaux d'horloge d'échantillonnage dans toute la fonction audio. Les fréquences d'échantillonnage au sein de la fonction audio ne peuvent être influencées que par une interaction directe avec la commande de fréquence d'échantillonnage au sein d'une entité source d'horloge. Les attributs RANGE de la commande de fréquence d'échantillonnage fournissent les informations nécessaires au logiciel hôte pour déterminer les fréquences d'échantillonnage prises en charge par la commande (et donc le domaine d'horloge associé).

Un effet secondaire de la modification de la fréquence d'échantillonnage peut être que certaines interfaces AudioStreaming peuvent avoir besoin de passer à un paramètre alternatif différent pour prendre en charge la bande passante nécessaire à la nouvelle fréquence d'échantillonnage. La présente spécification ne permet pas à une interface AudioStreaming de passer d'un paramètre alternatif à un autre d'elle-même, sauf pour passer au paramètre alternatif zéro, qui est le paramètre inactif. Au lieu de cela, lorsque la fonction audio détecte qu'elle ne peut plus prendre en charge un paramètre alternatif donné sur une interface AudioStreaming, elle doit basculer vers le paramètre alternatif zéro sur cette interface et signaler la modification au logiciel hôte à travers l'interruption de la commande de paramètre alternatif actif. L'hôte peut ensuite interroger l'interface pour obtenir de nouveaux paramètres alternatifs valides pour l'interface au moyen de la demande "Get Valid Alternate Settings Control" et effectuer un choix approprié.

Note: Pour conserver un nombre minimal de paramètres alternatifs dans une interface AudioStreaming, il est recommandé de ne pas fournir un paramètre alternatif distinct pour chaque fréquence d'échantillonnage prise en charge. Quelques paramètres alternatifs actifs (faible bande passante, bande passante moyenne, large bande passante) peuvent suffire à fournir un contrôle raisonnable de la bande passante.

Les flux audio peuvent être pontés d'un domaine d'horloge à l'autre en utilisant l'unité de conversion du débit d'échantillonnage.

3.14.4 MODELE DE DOMAINES D'ALIMENTATION

La prise en charge de la gestion des domaines d'alimentation est facultative pour une fonction audio. Si elle est prise en charge, la fonction audio est subdivisée en un ou plusieurs domaines d'alimentation, identifiés par leur ID de domaine d'alimentation unique, qui peuvent être manipulés individuellement par l'hôte afin d'optimiser la consommation globale d'énergie. Si elle est prise en charge, l'interface AudioControl contient une matrice de commandes de domaine d'alimentation qui permet à l'hôte de définir des états de domaine d'alimentation pour chaque domaine d'alimentation individuel au sein de la fonction audio. La matrice de commandes de domaine d'alimentation contient autant d'éléments qu'il y a de domaines d'alimentation et l'accès à une commande de domaine d'alimentation spécifique est fondé sur l'ID de domaine d'alimentation.

Un domaine d'alimentation doit prendre en charge trois états de domaine d'alimentation possibles, de D0 à D2. L'état de domaine d'alimentation D0 est l'état pleinement opérationnel et doit être l'état de domaine d'alimentation par défaut pour tous les domaines d'alimentation. L'état de domaine d'alimentation D1 est un état non fonctionnel qui doit consommer moins d'énergie que l'état de domaine d'alimentation D0, mais qui présente toujours la capacité de générer des événements de réveil (interruptions) pour les commandes de son domaine d'alimentation. L'état de domaine d'alimentation D2 doit consommer encore moins d'énergie que l'état de domaine d'alimentation D1 (éventuellement aucune), et ne doit être capable de générer aucun événement de réveil pour les commandes de son domaine d'alimentation.

Une fonction audio n'est autorisée à modifier l'état de domaine d'alimentation d'aucun de ses domaines d'alimentation de manière autonome. Un changement d'état de domaine d'alimentation doit toujours être initié par une demande explicite de l'hôte. Les états de domaine d'alimentation doivent être conservés à l'arrêt de la fonction ou du dispositif.

Note: Les états de domaine d'alimentation fonctionnent indépendamment de l'état d'interruption de la fonction ou du dispositif. Par exemple, en plaçant tous les domaines d'alimentation d'une fonction audio dans l'état de domaine d'alimentation D2 ne met pas automatiquement la fonction audio à l'arrêt.

Les niveaux de consommation d'énergie réels pour chaque état de domaine d'alimentation ne sont pas annoncés. La seule exigence est qu'un état de domaine d'alimentation de numérotation supérieure consomme moins d'énergie que tous les états de domaine d'alimentation de numérotation inférieure, potentiellement au détriment de temps de récupération plus élevés. Le temps de récupération est défini comme le temps approximativement nécessaire pour revenir à l'état de domaine d'alimentation pleinement opérationnel D0 à partir de l'état de domaine d'alimentation inférieur D1 ou D2. Le descripteur de domaine d'alimentation doit indiquer le temps de récupération pour passer de l'état de domaine d'alimentation D1 à D0 et de l'état de domaine d'alimentation D2 à D0.

La fonction audio est la mieux placée pour gérer les détails de ses ressources et de leur consommation d'énergie sous diverses conditions. Ces détails ne sont donc pas exposés à l'hôte. Au lieu de cela, l'hôte indique à la fonction audio quelles parties de la fonction audio il souhaite utiliser à un moment donné et la fonction audio doit décider de manière autonome quelles ressources elle peut en toute sécurité faire passer à un état d'alimentation inférieur.

À quelques exceptions près, l'hôte expose à la fonction audio la manière dont il va utiliser ses blocs de construction en indiquant les terminaux d'entrée et de sortie qu'il va activement utiliser pour répondre à un certain scénario d'utilisation. À partir de ces informations, la fonction audio peut déduire d'elle-même quelles ressources internes (entités) peuvent être commutées vers un état d'alimentation inférieur voire complètement arrêtées pour ce scénario d'utilisation. Par conséquent, il n'est pas nécessaire d'énumérer de manière exhaustive l'appartenance à un certain domaine d'alimentation pour toutes les entités de la fonction audio. Il suffit d'indiquer l'appartenance à un domaine d'alimentation pour les terminaux d'entrée et de sortie uniquement. En manipulant l'état d'un domaine d'alimentation en particulier, l'hôte indique à présent à la fonction audio s'il souhaite utiliser les entrées et/ou les sorties associées à ce domaine d'alimentation dans le scénario d'utilisation actuel.

Il est possible d'avoir un domaine d'alimentation qui ne comprend que des entités autres que des terminaux d'entrée et de sortie. Dans ce cas, il convient qu'un contournement explicite de ces entités soit mis en œuvre dans la topologie de la fonction audio au moyen d'une unité de sélection résidant en dehors de ce domaine d'alimentation. Le réglage de l'état de domaine d'alimentation du domaine d'alimentation auquel appartient une telle entité sur une autre valeur que D0 doit rendre cette entité non fonctionnelle et, par conséquent, elle doit cesser de produire une puissance significative (la puissance n'est pas définie par la présente spécification dans ce cas). Pour préserver l'intégrité du chemin de signal, le logiciel hôte a d'abord besoin de contourner l'entité en question (en choisissant l'entrée appropriée sur l'unité de sélection) avant de faire passer le domaine d'alimentation à un état d'alimentation inférieur.

Les entités qui peuvent être explicitement contournées peuvent être l'unique membre d'un domaine d'alimentation ou peuvent être un membre parmi d'autres, à condition que tous les membres qui ne sont pas des terminaux puissent être contournés soit individuellement, soit en tant que groupe.

3.14.5 CONSIDERATIONS ET EXIGENCES SUPPLEMENTAIRES RELATIVES A L'ALIMENTATION

Un dispositif alimenté par bus conforme Audio 3.0 doit prendre en charge LPM/L1 conformément aux exigences définies dans les errata USB 2.0 LPM/L1 ECN et LPM. Le sous-système USB du dispositif Audio 3.0 doit mettre en œuvre et entrer en mode basse puissance pour L1. Un dispositif audio 3.0 doit accepter les demandes d'entrée LPM/L1 (c'est-à-dire retourner un ACK) à tout moment, même si la mémoire tampon du point d'extrémité RX isochrone du dispositif n'a pas encore été vidée ou que le point d'extrémité TX isochrone contient déjà des échantillons pour la prochaine salve. L'hôte est responsable de la reprise du dispositif à temps pour desservir la prochaine salve isochrone et doit être fiable.

La seule exception se présente lorsque le dispositif a besoin de jetons uSOF supplémentaires afin de resynchroniser son horloge interne avec l'horloge USB. Dans ce cas, le dispositif peut refuser la demande de placer la liaison dans un état L1. Il ne doit le faire qu'une fois toutes les 64 ms ou plus.

Un hôte USB prenant en charge LPM/L1 est tenu de respecter les exigences définies dans les errata USB 2.0 LPM/L1 ECN et LPM. Toutefois, pour les points d'extrémité isochrones, l'hôte doit également être tenu d'envoyer au moins un jeton uSOF avant la reprise des transactions sur un point d'extrémité. L'hôte USB doit être tenu de demander une entrée LPM/L1 après la desserte du dispositif à chaque intervalle de service. L'hôte USB est en outre tenu de mettre en œuvre un état de puissance faible et d'économiser de l'énergie pendant L1.

3.14.6 LIAISON ENTRE LES BOUTONS PHYSIQUES ET LES COMMANDES AUDIO

La plupart des dispositifs comprenant une fonction audio disposent également d'un ou de plusieurs boutons de panneau avant destinés à contrôler certains aspects de la fonction audio au sein du dispositif. L'exemple le plus évident est celui d'un bouton de commande de volume sur le devant d'un haut-parleur multimédia. Etant donné qu'une fonction audio peut potentiellement contenir de nombreuses commandes audio du même type, il est nécessaire de lier une commande physique (bouton, poignée, curseur, manette, etc.) à une commande audio particulière au sein de la fonction audio.

La présente spécification fournit deux méthodes mutuellement exclusives pour fournir cette liaison:

- Le bouton physique est mis en œuvre en tant que commande HID
- Le bouton physique fait partie intégrante de la commande audio

Il est interdit de mettre en œuvre les deux méthodes pour un même bouton physique. Toutefois, il est admis d'utiliser la première méthode pour certains boutons du panneau avant et la seconde méthode pour les autres boutons du panneau avant. Il est fortement déconseillé de mettre en œuvre des boutons de panneau avant qui n'utilisent aucune des deux méthodes susmentionnées, c'est-à-dire des boutons qui sont invisibles pour le logiciel hôte et n'ont qu'un effet local.

3.14.6.1 LE BOUTON PHYSIQUE EST UNE COMMANDE HID

Dans ce cas, le bouton physique est complètement séparé de la fonction audio et est mis en œuvre au sein de l'interface HID du dispositif. La fonction audio n'a pas même connaissance de l'existence du bouton. Tout changement d'état du bouton est communiqué au logiciel hôte au moyen de rapports HID. Il revient alors au logiciel hôte d'interpréter le changement d'état du bouton et d'en déduire l'action appropriée à entreprendre sur la fonction audio. Par conséquent, la responsabilité de la liaison réside entièrement dans l'application ou dans le logiciel du système d'exploitation. Bien que cette méthode offre une grande flexibilité, elle place également la responsabilité de fournir la liaison correcte sur le logiciel, ce qui rend parfois difficile la création d'applications génériques ou de logiciels de système d'exploitation qui génèrent la liaison appropriée (prévue par le fabricant).

3.14.6.2 LE BOUTON PHYSIQUE FAIT PARTIE INTEGRANTE DE LA COMMANDE AUDIO

Dans ce cas, le bouton physique interagit directement avec la commande audio réelle. Les changements d'état du bouton ne sont pas signalés au logiciel hôte. Au lieu de cela, le changement d'état de la commande audio résultant de la manipulation du bouton est signalé au logiciel hôte par le biais du mécanisme d'interruption de la commande audio. Par conséquent, la liaison entre le bouton physique et la commande audio est très directe et entièrement dictée par la conception du dispositif. Bien que moins flexible, cette méthode offre une manière très claire et directe d'effectuer la liaison.

4 DESCRIPTEURS

Les sections suivantes décrivent les descripteurs USB normalisés et spécifiques à la classe pour la classe d'interface audio.

4.1 DESCRIPTEURS NORMALISES

Étant donné que la fonctionnalité audio est toujours réputée résider au niveau de l'interface, tous les champs pertinents des descripteurs normalisés doivent indiquer que les informations de classe doivent se trouver au niveau de l'interface, de sorte que le logiciel d'énumération regarde au niveau de l'interface pour déterminer la classe d'interface et pour aussi s'assurer que le logiciel d'énumération prenant en compte le descripteur d'association d'interfaces est chargé.

Étant donné qu'il est exigé que tout dispositif conforme à la présente spécification contienne plusieurs descripteurs de configuration, il est également exigé que ces dispositifs incluent un descripteur de synthèse de configuration dans le descripteur BOS normalisé. Ces descripteurs de synthèse de configuration peuvent être utilisés par le logiciel hôte pour décider quelle configuration utiliser pour obtenir la fonctionnalité souhaitée, non seulement pour une fonction audio, mais également pour toute autre fonction disponible dans la configuration. La partie fonction audio du descripteur de synthèse de configuration doit contenir les champs **bFunctionClass**, **bFunctionSubClass** et **bFunctionProtocol** de chaque AIA contenue dans la configuration synthétisée. Comme mentionné précédemment en 3.3, "Rétrocompatibilité", il peut être nécessaire d'inclure la même AIA dans plusieurs descripteurs de configuration si le dispositif USB contient d'autres fonctions de classe USB avec des versions différentes.

Pour plus d'informations, se référer à la synthèse de configuration BOS ECR/ECN.

Noter également que, comme il est exigé que les dispositifs conformes à la présente spécification contiennent un descripteur BOS, la valeur minimale de **bDeviceProtocol** dans le descripteur de dispositif USB normalisé doit être 0x0201.

Le mécanisme d'association d'interfaces normalisé est utilisé pour décrire une association d'interfaces audio (AIA). Toutes les interfaces appartenant à la même AIA doivent être identifiées au moyen du descripteur d'association d'interfaces (IAD) normalisé.

Chaque AIA doit comprendre l'interface AudioControl obligatoire qui doit être la première de l'AIA (celle ayant le numéro d'interface le plus faible). Toutes les interfaces AudioStreaming doivent être numérotées de façon contiguë et suivre immédiatement l'interface AudioControl dans l'AIA. Toutes les interfaces MIDIStreaming doivent être numérotées de façon contiguë et suivre immédiatement les interfaces AudioStreaming dans l'AIA.

Note: Pour plus d'informations sur l'association d'interfaces, se référer au livre blanc *USB Interface Association Descriptor Device Class Code and Use Model White Paper*, disponible sur le site web de l'USB.

4.2 DESCRIPTEURS SPECIFIQUES A LA CLASSE

La présente spécification admet deux méthodes différentes pour exprimer les descripteurs spécifiques à la classe. La première méthode suit la topologie traditionnelle des descripteurs USB, tandis que la seconde méthode introduit une nouvelle façon d'exprimer et de récupérer les descripteurs spécifiques à la classe. Ces descripteurs sont appelés descripteurs haute capacité. Pour chaque descripteur spécifique à la classe, les mises en œuvre doivent utiliser la méthode indiquée dans la présente spécification. Les descripteurs haute capacité sont utilisés dans les cas suivants:

- La longueur du descripteur dépasse la limite de 256 octets imposée par les descripteurs traditionnels

- Le descripteur est de nature dynamique selon la présente spécification, c'est-à-dire qu'il peut être modifié après l'énumération (les descripteurs haute capacité sont capables de signaler les modifications).

Les descripteurs haute capacité ne font jamais partie de la hiérarchie des descripteurs de configuration, renvoyés par la demande "Get Configuration". Seuls les descripteurs avec une topologie traditionnelle peuvent être inclus dans cette hiérarchie. Un descripteur haute capacité doit toujours être référencé par un descripteur traditionnel spécifique à la classe qui inclut l'ID unique du descripteur haute capacité comme l'un de ses champs.

4.2.1 DESCRIPTEURS TRADITIONNELS SPECIFIQUES A LA CLASSE

Les descripteurs traditionnels spécifiques à la classe définis dans la présente spécification suivent tous une topologie commune. Les trois premiers champs d'un descripteur traditionnel spécifique à la classe sont communs à tous les descripteurs traditionnels et sont suivis d'une topologie spécifique au type et au sous-type du descripteur.

Le champ **bLength** indique la longueur totale du descripteur, en octets.

Le champ **bDescriptorType** suit en partie le schéma d'allocation des bits du champ **bmRequestType** et identifie le descripteur comme étant un descripteur spécifique à la classe. Le bit D7 de ce champ est réservé. Les bits D6..5 sont utilisés pour indiquer qu'il s'agit d'un descripteur spécifique à la classe (D6..5 = 0b01). Les bits D4..0 sont utilisés pour coder le type de descripteur.

Le champ **bDescriptorSubtype** qualifie plus précisément la nature exacte du descripteur.

Tableau 4-1: Topologie des descripteurs traditionnels spécifiques à la classe

Décalage	Champ	Taille	Valeur	Description
0	bLength	1	Nombre	Taille de ce descripteur, en octets: 3+n.
1	bDescriptorType	1	0b001xxxx	Type de descripteur:
2	bDescriptorSubtype	1	Constante	Sous-type de descripteur.
3	---	n	---	Topologie du descripteur spécifique au type et au sous-type du descripteur.

4.2.1.1 CHAMPS COMMUNS DANS CERTAINS DESCRIPTEURS SPECIFIQUES A LA CLASSE

Certains champs apparaissent dans plusieurs descripteurs spécifiques à la classe. Au lieu de répéter la description de ce champ pour toutes les instances de descripteurs, la description est indiquée une fois dans la présente section et s'applique à toutes les instances de descripteurs dans lesquelles le champ apparaît.

4.2.1.1.1 CHAMP BMCONTROLS

Le champ **bmControls** contient un ensemble de paires de bits, indiquant quelles commandes sont présentes dans l'entité et quelles sont leurs capacités. Si une certaine commande n'est pas présente, la paire de bits doit être définie sur 0b00. Si une commande est présente, mais en lecture seule, la paire de bits doit être définie sur 0b01. Si une commande est également programmable par l'hôte, la paire de bits doit être définie sur 0b11 (lecture/écriture). La valeur 0b10 n'est pas admise.

4.2.2 DESCRIPTEURS HAUTE CAPACITE SPECIFIQUES A LA CLASSE

La présente spécification définit un nouveau type de descripteurs spécifiques à la classe, appelés descripteurs haute capacité. Un descripteur haute capacité n'est pas récupéré à partir de la fonction audio au moment de l'énumération. Au lieu de cela, il ne peut être récupéré qu'à l'aide de la demande "Get High Capability Descriptor" définie en 5.2.3.3, "Demande de descripteur haute capacité".

Un descripteur haute capacité est toujours référencé par un autre descripteur au moyen de son ID unique.

Un descripteur haute capacité spécifique à la classe peut avoir une longueur allant jusqu'à 64K octets. En outre, il peut générer une interruption de type source HIGH_CAPABILITY_DESCRIPTOR chaque fois que la fonction audio modifie le descripteur en fonctionnement normal (descripteur dynamique).

Les descripteurs haute capacité suivent tous une topologie commune, très semblable à celle des descripteurs traditionnels spécifiques à la classe. Les trois premiers champs d'un descripteur haute capacité sont communs à tous les descripteurs haute capacité et sont suivis d'une topologie spécifique au type et au sous-type du descripteur.

Le champ **wLength** indique la longueur totale du descripteur, en octets. Les longueurs de descripteurs allant jusqu'à 64K octets sont prises en charge.

Le champ **bDescriptorType** suit en partie le schéma d'allocation des bits du champ **bmRequestType** et identifie le descripteur comme étant un descripteur spécifique à la classe. Le bit D7 de ce champ est réservé. Les bits D6..5 sont utilisés pour indiquer qu'il s'agit d'un descripteur spécifique à la classe (D6..5 = 0b01). Les bits D4..0 sont utilisés pour coder le type de descripteur.

Le champ **bDescriptorSubtype** qualifie plus précisément la nature exacte du descripteur. Les bits D6..0 du champ **bDescriptorSubtype** sont utilisés pour coder le sous-type de descripteur.

Le champ **wDescriptorID** contient une valeur qui identifie de manière unique le descripteur haute capacité au sein d'une interface ou d'un point d'extrémité.

Tableau 4-2: Topologie des descripteurs haute capacité spécifiques à la classe

Décalage	Champ	Taille	Valeur	Description
0	wLength	2	Nombre	Taille de ce descripteur, en octets: 6+n.
2	bDescriptorType	1	0b001xxxxx	Type de descripteur:
3	bDescriptorSubtype	1	Constante	Sous-type de descripteur.
4	wDescriptorID	2	Nombre	ID unique pour ce descripteur haute capacité spécifique à la classe.
6	---	n	---	Topologie du descripteur spécifique au type et au sous-type du descripteur.

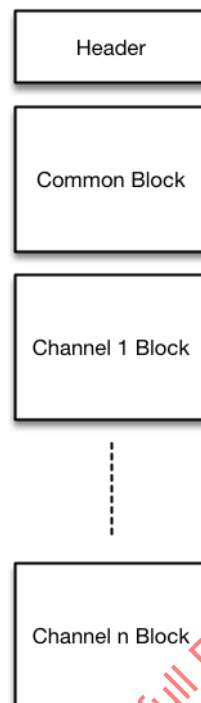
4.3 DESCRIPTEUR DE GRAPPE

Une grappe est un regroupement de canaux audio qui partagent les mêmes caractéristiques, comme la fréquence d'échantillonnage, la résolution de bit, etc. Pour caractériser une grappe, un descripteur de grappe est introduit.

La présente spécification introduit un descripteur de grappe qui est significativement différent des définitions de descripteurs de grappe précédentes. L'objectif est d'être beaucoup plus flexible et extensible tout en décrivant la grappe plus précisément et avec un contenu plus riche.

Le descripteur de grappe utilise toujours la représentation haute capacité. Au niveau le plus élevé, il est structuré comme suit:

Figure 4-1: Descripteur de grappe



Anglais	Français
Header	En-tête
Channel 1 Block	Bloc canal 1
Channel n Block	Bloc canal n
Common Block	Bloc commun

Le descripteur est constitué d'un en-tête fixe, suivi d'un bloc commun facultatif, suivi d'autant de blocs de canal qu'il y a de canaux dans la grappe.

4.3.1 EN-TÊTE DE DESCRIPTEUR DE GRAPPE

L'en-tête commence par le champ **wLength** qui contient le nombre total d'octets dans tout le descripteur.

Le champ **wDescriptorID** contient un numéro ID qui identifie de manière unique le descripteur au sein de la fonction audio. La valeur zéro est réservée et ne doit pas être utilisée comme ID de descripteur de grappe valide.

Le champ **bNrChannels** indique le nombre de canaux audio présents dans la grappe.

Tableau 4-3: En-tête de descripteur de grappe

Décalage	Champ	Taille	Valeur	Description
0	wLength	2	Nombre	Longueur totale du descripteur de grappe, en octets.
2	bDescriptorType	1	Constante	Type de descripteur CS_CLUSTER.
3	bDescriptorSubtype	1	Constante	Sous-type de descripteur SUBTYPE_UNDEFINED.
4	wDescriptorID	2	Nombre	ID unique de ce descripteur de grappe.
6	bNrChannels	1	Nombre	Nombre de canaux présents dans la grappe: n.

4.3.2 BLOC DE DESCRIPTEUR DE GRAPPE

L'en-tête du descripteur de grappe est suivi d'un certain nombre de blocs de descripteur de grappe. Il y a un bloc commun facultatif, suivi d'autant de blocs de canal qu'il y a de canaux dans la grappe. Chaque bloc est composé d'un certain nombre de segments.

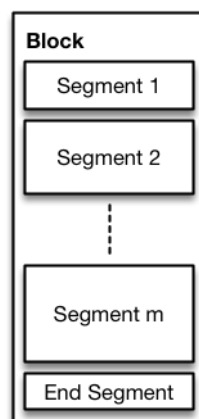
Le bloc commun est composé de segments qui contiennent des informations pertinentes sur les caractéristiques de la grappe dans son ensemble.

Un bloc de canal est composé de segments qui contiennent des informations pertinentes sur les caractéristiques de ce canal. Au minimum, chaque bloc de canal doit contenir soit un segment d'informations, soit un segment d'ambiphonie ainsi qu'un segment de fin unique. Tous les autres segments sont facultatifs. Il est fortement recommandé d'utiliser la même topologie pour chaque bloc de canal, c'est-à-dire que les mêmes segments apparaissent dans le même ordre dans chaque bloc de canal.

Toutefois, si, pour une raison quelconque, il n'existe actuellement aucune information de grappe disponible, le descripteur de grappe ne doit contenir qu'un en-tête dont le champ **bNrChannels** est défini sur zéro. Il s'agit du descripteur de grappe vide.

La Figure 4-2 représente plus en détail les concepts ci-dessus.

Figure 4-2: Bloc de descripteur de grappe



Anglais	Français
Block	Bloc
End Segment	Segment de fin

4.3.2.1 SEGMENTS

Il existe deux types de segments: Les segments de bloc commun contiennent des informations pertinentes sur la grappe dans son ensemble. Les segments de bloc de canal contiennent des informations pertinentes sur certains aspects d'un canal en particulier. Les deux types de segments partagent la même topologie.

Le champ **bSegmentType** décrit le type de segment (bloc commun ou bloc de canal) ainsi que le type de contenu dans le segment.

La topologie d'un segment est toujours la suivante:

Tableau 4-4: Segment de descripteur de grappe

Décalage	Champ	Taille	Valeur	Description
0	wLength	2	Nombre	Longueur du segment, en octets: 3+n.
2	bSegmentType	1	Constante	Décrit le type de segment et le type de contenu du segment.
3	Spécifique au segment	n		Contenu spécifique au segment.

4.3.2.1.1 SEGMENT DE FIN

Chaque bloc se termine par un segment de fin. Le segment de fin marque la fin du bloc à longueur variable. Le segment de fin ne comporte pas de section spécifique au segment et est structuré comme suit:

Tableau 4-5: Segment de fin

Décalage	Champ	Taille	Valeur	Description
0	wLength	2	Nombre	Longueur du segment de fin, en octets: 3.
2	bSegmentType	1	Constante	END_SEGMENT.

4.3.2.1.2 SEGMENTS DE BLOCS COMMUNS

Les types de segments de blocs communs suivants sont définis:

- Description de grappe
- Spécifique au fournisseur

Les valeurs des types de segments de bloc commun peuvent être consultées en A.10, "Types de segments de descripteurs de grappes".

4.3.2.1.2.1 SEGMENT DE DESCRIPTION DE GRAPPE

Le segment de description de grappe contient l'ID d'un descripteur de chaîne dans le champ **wCDDescrStr** qui fournit plus d'informations sur la grappe.

Tableau 4-6: Segment de description de grappe

Décalage	Champ	Taille	Valeur	Description
0	wLength	2	Nombre	Longueur du segment, en octets: 5.
2	bSegmentType	1	Constante	CLUSTER_DESCRIPTION.
3	wCDDescrStr	2	wStrDescrID	ID d'un descripteur de chaîne qui décrit plus précisément la grappe.

4.3.2.1.2.2 SEGMENT DEFINI PAR LE FOURNISSEUR

Les fournisseurs sont autorisés à ajouter des segments définis par le fournisseur au bloc commun afin de transmettre des informations propriétaires supplémentaires sur la grappe. Le segment défini par le fournisseur doit utiliser la topologie suivante:

Tableau 4-7: Segment défini par le fournisseur

Décalage	Champ	Taille	Valeur	Description
0	wLength	2	Nombre	Longueur du segment de fin, en octets: 3+n.
2	bSegmentType	1	Constante	CLUSTER_VENDOR_DEFINED.
3	Défini par le fournisseur	n	N/A	Extension du segment définie par le fournisseur.

4.3.2.1.3 SEGMENTS DE BLOCS DE CANAUX

Les types de segments de blocs de canaux suivants sont définis:

- Informations
- Ambiphonie
- Description du canal
- Spécifique au fournisseur

Les valeurs des types de segments de blocs de canaux peuvent être consultées à l'Annexe A.10, "Types de segments de descripteurs de grappes".

4.3.2.1.3.1 SEGMENT D'INFORMATIONS

Le segment d'informations contient des informations pertinentes pour un canal particulier de la grappe. Il est mutuellement exclusif avec le segment d'ambiphonie pour ce même canal.

Le champ **bChPurpose** indique l'objectif du canal. Les objectifs actuellement définis pour un canal sont les suivants:

- Generic Audio: contient de l'audio principalement utilisé pour la capture ou la reproduction directe.
- Voice: destiné à être interprété par les humains.
- Speech: destiné à être interprété ou généré par une machine.
- Ambient: contient de l'audio autre que celui des canaux principaux.
- Reference: contient l'audio traité final. Par exemple, une référence pour le traitement AEC.
- Ultrasonic: contient des signaux dont le contenu spectral dépasse les limites audibles (généralement >20 kHz).
- Vibrokinetic: contient des informations très basses fréquences, généralement utilisées pour actionner les vibrateurs ou les actionneurs de mouvement.
- Non-Audio: indique que le canal transporte des informations non audio. Les données de détection de pression en temps réel ou les données de rétroaction de gain d'amplificateur, etc. sont des exemples d'informations non audio.

Les valeurs pour les objectifs répertoriés ci-dessus peuvent être consultées en A.11, "Définitions des objectifs des canaux".

Noter qu'il convient de faire très attention lors de l'indication des objectifs d'un canal audio. Ce n'est que lorsque le canal a été spécifiquement ajusté pour répondre à un objectif qu'il convient de définir le ou les bits appropriés. Par exemple, si l'audio brut est traité spécifiquement pour être utilisé par des applications qui utilisent la voix humaine comme entrée, le canal peut être marqué "Voice".

Le champ **wChRelationship** décrit la relation de canal avec les autres canaux de la grappe. Les relations actuellement définies pour un canal sont décrites dans le tableau ci-dessous.

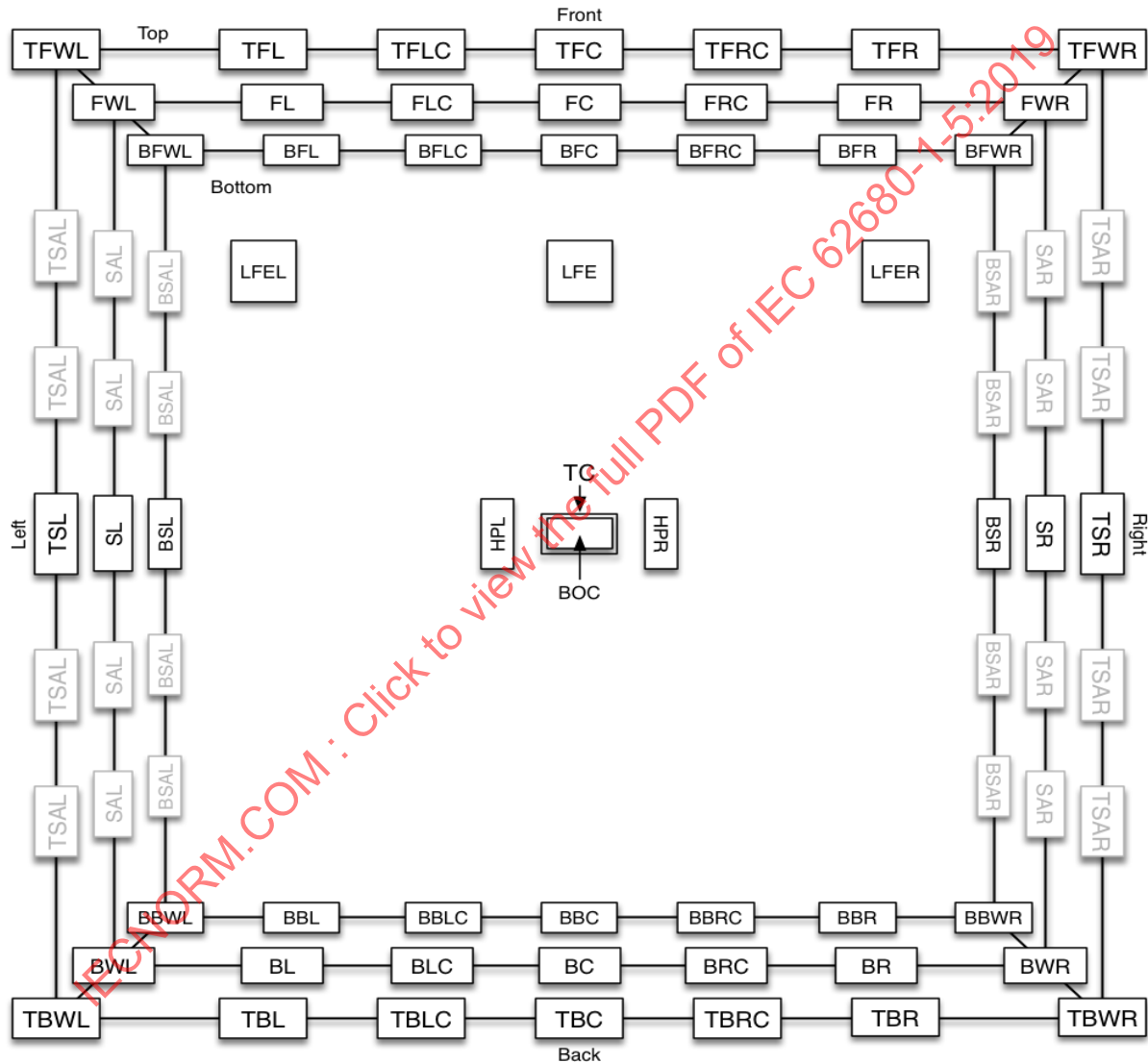
Tableau 4-8: Relations des canaux

	Relation du canal	Acronyme	Description
Générique	RELATIONSHIP_UNDEFINED	UND	Relation non définie ou inconnue
	MONO	M	Canal mono
	LEFT	L	Canal gauche générique
	RIGHT	R	Canal droit générique
	ARRAY	AR	Canal faisant partie d'une configuration de réseau
Lié à l'entrée	PATTERN_X	PX	Canal X d'un microphone X/Y
	PATTERN_Y	PY	Canal Y d'un microphone X/Y
	PATTERN_A	PA	Canal A d'un microphone A/B
	PATTERN_B	PB	Canal B d'un microphone A/B
	PATTERN_M	PM	Canal M d'un microphone M/S
	PATTERN_S	PS	Le canal S d'un microphone M/S
Lié à la sortie	Avant gauche	FL	Voir Figure 4-3
	Avant droite	FR	Voir Figure 4-3
	Centre avant	FC	Voir Figure 4-3
	Centre avant gauche	FLC	Voir Figure 4-3
	Centre avant droite	FRC	Voir Figure 4-3
	Avant gauche large	FWL	Voir Figure 4-3
	Avant droite large	FWR	Voir Figure 4-3
	Côté gauche	SL	Voir Figure 4-3
	Côté droite	SR	Voir Figure 4-3
	Réseau surround gauche	SAL	Voir Figure 4-3
	Réseau surround droite	SAR	Voir Figure 4-3
	Arrière gauche	BL	Voir Figure 4-3
	Arrière droite	BR	Voir Figure 4-3
	Centre arrière	BC	Voir Figure 4-3
	Centre arrière gauche	BLC	Voir Figure 4-3
	Centre arrière droite	BRC	Voir Figure 4-3
	Arrière gauche large	BWL	Voir Figure 4-3
	Arrière droite large	BWR	Voir Figure 4-3
	Centre supérieur	TC	Voir Figure 4-3
	Avant supérieur gauche	TFL	Voir Figure 4-3
	Avant supérieur droite	TFR	Voir Figure 4-3
	Centre supérieur avant	TFC	Voir Figure 4-3
	Centre supérieur avant gauche	TFLC	Voir Figure 4-3
	Centre supérieur avant droite	TFRC	Voir Figure 4-3

	Relation du canal	Acronyme	Description
	Avant supérieur gauche large	TFWL	Voir Figure 4-3
	Avant supérieur droite large	TFWR	Voir Figure 4-3
	Côté supérieur gauche	TSL	Voir Figure 4-3
	Côté supérieur droite	TSR	Voir Figure 4-3
	Réseau surround supérieur gauche	TSAL	Voir Figure 4-3
	Réseau surround supérieur droite	TSAR	Voir Figure 4-3
	Arrière supérieur gauche	TBL	Voir Figure 4-3
	Arrière supérieur droite	TBR	Voir Figure 4-3
	Centre arrière supérieur	TBC	Voir Figure 4-3
	Centre arrière supérieur gauche	TBLC	Voir Figure 4-3
	Centre arrière supérieur droite	TBRC	Voir Figure 4-3
	Arrière supérieur gauche large	TBWL	Voir Figure 4-3
	Arrière supérieur droite large	TBWR	Voir Figure 4-3
	Centre inférieur	BC	Voir Figure 4-3
	Avant inférieur gauche	BFL	Voir Figure 4-3
	Avant inférieur droite	BFR	Voir Figure 4-3
	Centre inférieur avant	BFC	Voir Figure 4-3
	Centre inférieur avant gauche	BFLC	Voir Figure 4-3
	Centre inférieur avant droite	BFRC	Voir Figure 4-3
	Avant inférieur gauche large	BFWL	Voir Figure 4-3
	Avant inférieur droite large	BFWR	Voir Figure 4-3
	Côté inférieur gauche	BSL	Voir Figure 4-3
	Côté inférieur droite	BSR	Voir Figure 4-3
	Réseau surround inférieur gauche	BSAL	Voir Figure 4-3
	Réseau surround inférieur droite	BSAR	Voir Figure 4-3
	Arrière inférieur gauche	BBL	Voir Figure 4-3
	Arrière inférieur droite	BBR	Voir Figure 4-3
	Centre inférieur arrière	BBC	Voir Figure 4-3
	Centre inférieur arrière gauche	BBLC	Voir Figure 4-3
	Centre inférieur arrière droite	BBRC	Voir Figure 4-3
	Arrière inférieur gauche large	BBWL	Voir Figure 4-3
	Arrière inférieur droite large	BBWR	Voir Figure 4-3
	Effets basses fréquences	LFE	Voir Figure 4-3

Relation du canal	Acronyme	Description
Effets basses fréquences gauche	LFEL	Voir Figure 4-3
Effets basses fréquences droite	LFER	Voir Figure 4-3
Casque gauche	HPL	Voir Figure 4-3
Casque droit	HPR	Voir Figure 4-3

Figure 4-3: Représentation en 3D des relations des canaux



Anglais	Français
Top	Haut
Front	Avant
Left	Gauche
Right	Droite
Bottom	Bas
Back	Arrière

Les noms de constantes, les acronymes et leurs valeurs associées sont répertoriés en A.12, "Définitions des relations de canaux".

Le champ **bChGroupID** permet de créer une liaison dans un sous-ensemble de canaux de la grappe en spécifiant la même valeur dans le champ **bChGroupID** pour ces canaux. Ce champ est utile lorsque plusieurs ensembles de canaux liés sont présents dans la même grappe. Par exemple, une grappe unique peut transporter un groupe de canaux 5.1 et un autre groupe de canaux microphones. Les canaux 5.1 partagent la même valeur d'ID de groupe, indiquant qu'ils appartiennent à un groupe (5.1), et les canaux microphones partagent une autre valeur d'ID de groupe, indiquant qu'ils appartiennent à un groupe différent (microphone). Le domaine d'application du champ **bChGroupID** est limité à la grappe à laquelle les canaux appartiennent. Noter que la spécification d'une valeur de 0 dans ce champ pour tous les canaux de la grappe crée effectivement un groupe avec ID=0 auquel tous les canaux appartiennent.

Tableau 4-9: Segment d'informations

Décalage	Champ	Taille	Valeur	Description
0	wLength	2	Nombre	Longueur du segment d'informations du canal, en octets: 6.
2	bSegmentType	1	Constante	CHANNEL_INFORMATION.
3	bChPurpose	1	Nombre	Objectif prévu du canal.
4	bChRelationship	1	Nombre	Décrit la relation de ce canal avec les autres canaux. Voir A.12, "Définitions des relations de canaux".
5	bChGroupID	1	Nombre	ID utilisé pour regrouper des canaux.

4.3.2.1.3.2 SEGMENT D'AMBIOPHONIE

Le segment d'ambiophonie contient des informations pertinentes pour un canal d'ambiophonie particulier de la grappe. Il est mutuellement exclusif avec le segment d'informations pour ce même canal.

Le champ **bCompOrdering** indique la convention utilisée pour l'ordre des harmoniques sphériques. Voir A.13, "Types de conventions d'ordre des composants d'ambiophonie" pour les conventions d'ordre des composants prises en charge. Tous les canaux d'un groupe (voir ci-dessous) doivent indiquer la même convention d'ordre des composants.

Le champ **bACN** contient le numéro du canal d'ambiophonie.

Le champ **bAmbNorm** indique le type de normalisation utilisé pour le canal. Voir A.14, "Types de normalisations d'ambiophonie" pour les types de normalisations pris en charge. Tous les canaux d'un groupe (voir ci-dessous) doivent indiquer le même type de normalisation.

Le champ **bChGroupID** permet de créer une liaison dans un sous-ensemble de canaux de la grappe en spécifiant la même valeur dans le champ **bChGroupID** pour ces canaux. Ce champ est utile lorsque plusieurs ensembles de canaux liés sont présents dans la même grappe. Par exemple, une grappe unique peut transporter un groupe de canaux 5.1 et un autre groupe de canaux microphones. Les canaux 5.1 partagent la même valeur d'ID de groupe, indiquant qu'ils appartiennent à un groupe (5.1), et les canaux microphones partagent une autre valeur d'ID de groupe, indiquant qu'ils appartiennent à un groupe différent (microphone). Le domaine d'application du champ **bChGroupID** est limité à la grappe à laquelle les canaux appartiennent. Noter que la spécification d'une valeur de 0 dans ce champ pour tous les canaux de la grappe crée effectivement un groupe avec ID=0 auquel tous les canaux appartiennent.

Tableau 4-10: Segment d'ambiophonie

Décalage	Champ	Taille	Valeur	Description
0	wLength	2	Nombre	Longueur du segment d'informations du canal d'ambiophonie, en octets: 7.
2	bSegmentType	1	Constante	CHANNEL_AMBISONIC.
3	bCompOrdering	1	Nombre	Convention d'ordre des composants pour les harmoniques sphériques ambiophoniques. Voir A.13, "Types de conventions d'ordre des composants d'ambiophonie".
4	bACN	1	Nombre	Nombre de canaux d'ambiophonie.
5	bAmbNorm	1	Nombre	Normalisation des canaux appliquée. Voir A.14, "Types de normalisations d'ambiophonie".
6	bChGroupID	1	Nombre	ID utilisé pour regrouper des canaux.

4.3.2.1.3.3 SEGMENT DE DESCRIPTION DE CANAL

Le segment de description de canal contient l'ID d'un descripteur de chaîne dans le champ **wChDescrStr** qui fournit plus d'informations sur le canal.

Tableau 4-11: Segment de description de canal

Décalage	Champ	Taille	Valeur	Description
0	wLength	2	Nombre	Longueur du segment, en octets: 5.
2	bSegmentType	1	Constante	CHANNEL_DESCRIPTION.
3	wChDescrStr	2	wStrDescrID	ID d'un descripteur de chaîne qui décrit plus précisément le canal.

4.3.2.1.3.4 SEGMENT DÉFINI PAR LE FOURNISSEUR

Les fournisseurs sont autorisés à ajouter des segments définis par le fournisseur au bloc de canal afin de transmettre des informations propriétaires supplémentaires sur le canal. Le segment défini par le fournisseur doit utiliser la topologie suivante:

Tableau 4-12: Segment défini par le fournisseur

Décalage	Champ	Taille	Valeur	Description
0	wLength	2	Nombre	Longueur du segment de fin, en octets: 3+n.
2	bSegmentType	1	Constante	CHANNEL_VENDOR_DEFINED.
3	Défini par le fournisseur	n	N/A	Extension du segment définie par le fournisseur.

4.3.3 EXEMPLE DE DESCRIPTEUR DE GRAPPE

Une grappe qui transporte des canaux 5.1 distincts peut avoir le descripteur de grappe suivant:

Tableau 4-13: Exemple de descripteur de grappe

	Décalage	Champ	Taille	Valeur	Description
En-tête	0	wLength	2	0x3E	Longueur totale du descripteur de grappe, en octets: 62.
	2	bDescriptorType	1	0x26	Type de descripteur CS_CLUSTER.
	3	bDescriptorSubtype	1	0x00	Sous-type de descripteur SUBTYPE_UNDEFINED.
	4	wDescriptorID	2	En fonction de la mise en œuvre	ID unique de ce descripteur de grappe.
	6	bNrChannels	1	6	Nombre de canaux présents dans la grappe.
Segment d'informations	8	wLength	2	0x06	Longueur du segment d'informations, en octets.
	10	bSegmentType	1	0x20	CHANNEL_INFORMATION.
	11	bChPurpose	1	0x00	Audio générique.
	12	bChRelationship	1	0x80	Avant gauche.
	13	bChGroupID	1	0x01	ID utilisé pour regrouper des canaux.
Fin	14	wLength	2	0x03	Longueur du segment de fin, en octets.
	16	bSegmentType	1	0xFF	END_SEGMENT.
Informations	17	wLength	2	0x06	Longueur du segment d'informations, en octets.
	19	bSegmentType	1	0x20	CHANNEL_INFORMATION.
	20	bChPurpose	1	0x00	Audio générique.
	21	bChRelationship	1	0x81	Avant droite.
	22	bChGroupID	1	0x01	ID utilisé pour regrouper des canaux.
Fin	23	wLength	2	0x03	Longueur du segment de fin, en octets.
	25	bSegmentType	1	0xFF	END_SEGMENT.
Informations	26	wLength	2	0x06	Longueur du segment d'informations.
	28	bSegmentType	1	0x20	CHANNEL_INFORMATION.
	29	bChPurpose	1	0x00	Audio générique.
	30	bChRelationship	1	0x82	Centre avant.
	31	bChGroupID	1	0x01	ID utilisé pour regrouper des canaux.
Fin	32	wLength	2	0x03	Longueur du segment de fin, en octets.
	34	bSegmentType	1	0xFF	END_SEGMENT.
Informations	35	wLength	2	0x06	Longueur du segment d'informations.
	37	bSegmentType	1	0x20	CHANNEL_INFORMATION.
	38	bChPurpose	1	0x00	Audio générique.
	39	bChRelationship	1	0x89	Réseau surround gauche.
	40	bChGroupID	1	0x01	ID utilisé pour regrouper des canaux.
F	41	wLength	2	0x03	Longueur du segment de fin.

	Décalage	Champ	Taille	Valeur	Description
	43	bSegmentType	1	0xFF	END_SEGMENT.
Informations	44	wLength	2	0x06	Longueur du segment d'informations.
	46	bSegmentType	1	0x20	CHANNEL_INFORMATION.
	47	bChPurpose	1	0x00	Audio générique.
	48	bChRelationship	1	0x8A	Réseau surround droite.
	49	bChGroupID	1	0x01	ID utilisé pour regrouper des canaux.
Fin	50	wLength	2	0x03	Longueur du segment de fin.
	52	bSegmentType	1	0xFF	END_SEGMENT.
Informations	53	wLength	2	0x06	Longueur du segment d'informations.
	55	bSegmentType	1	0x20	CHANNEL_INFORMATION.
	56	bChPurpose	1	0x00	Audio générique.
	57	bChRelationship	1	0xB8	Effets basses fréquences.
	58	bChGroupID	1	0x01	ID utilisé pour regrouper des canaux.
Fin	59	wLength	2	0x03	Longueur du segment de fin.
	61	bSegmentType	1	0xFF	END_SEGMENT.

4.3.4 CEA-861.2 MISE EN CORRESPONDANCE DES CANAUX

La spécification d'extensions audio avancées CEA-861.2 affecte les localisations spatiales des haut-parleurs dans un ordre différent de celui défini dans la présente définition USB Audio. En outre, il existe des relations de canaux définies qui n'apparaissent pas dans la définition USB Audio. Pour assurer la cohérence des mises en œuvre utilisant à la fois les allocations CEA-861.2 et les relations de canaux audio USB, un schéma de mise correspondance entre les deux, auquel ces mises en œuvre doivent se conformer, est inclus ici.

La mise en correspondance des relations de canaux définies par USB et des allocations de haut-parleurs CEA est incluse dans le tableau donné en A.12, "Définitions des relations de canaux".

4.4 GRAPPE PHYSIQUE ET GRAPPE LOGIQUE

La présente spécification établit une distinction entre une grappe physique et une grappe logique. Par conséquent, deux types de descripteurs de grappe sont également définis:

- Descripteur de grappe logique
- Descripteur de grappe physique

La topologie de ces descripteurs est identique et doit toujours utiliser la représentation haute capacité décrite en 4.2.2, "Descripteurs haute capacité spécifiques à la classe". Les descripteurs de grappes logiques et physiques ne sont pas des descripteurs indépendants en tant que tels. Ils sont toujours référencés par d'autres descripteurs. Les descripteurs de référence incluent toujours un champ **wClusterDescrID** qui contient l'ID unique du descripteur de grappe auquel ils font référence.

Le descripteur de grappe logique est référencé par l'un des descripteurs suivants:

- Descripteur de terminal d'entrée
- Descripteur d'unité de mixage
- Descripteur d'unité de traitement
- Descripteur d'unité d'extension

Le descripteur de grappe physique est référencé par le descripteur d'interface AudioStreaming spécifique à la classe de chaque paramètre alternatif d'une interface AudioStreaming (à l'exception du paramètre alternatif 0).

Le descripteur de grappe physique est également référencé par le descripteur de connecteurs afin de fournir des informations sur les canaux physiques qui traversent les différents connecteurs, associés à un terminal.

4.4.1 MISE EN CORRESPONDANCE ENTRE GRAPPES PHYSIQUES ET LOGIQUES

Les flux audio entrent ou sortent toujours de la fonction audio au moyen d'une paire de terminaux AudioStreaming. Les sections suivantes décrivent plus en détail les cas possibles et les opérations internes qui interviennent.

4.4.1.1 INTERFACE AUDIOSTREAMING OUT – TERMINAL D'ENTREE

Pour les flux audio de Type I, le descripteur de grappe physique référencé par le descripteur d'interface AudioStreaming spécifique à la classe du paramètre alternatif actif de cette interface décrit les canaux audio qui entrent dans l'interface.

Pour les flux audio de Type III/IV, la grappe physique décrit les canaux audio après leur décodage en canaux audio distincts.

Les échantillons audio (sous-emplacements audio) doivent être ordonnés exactement de la même façon que dans le descripteur de grappe physique. La grappe entre ensuite dans un processus de mixage élévateur/réducteur implicite, afin de produire une grappe de sortie telle que décrite par le descripteur de grappe référencé par le descripteur de terminal d'entrée associé.

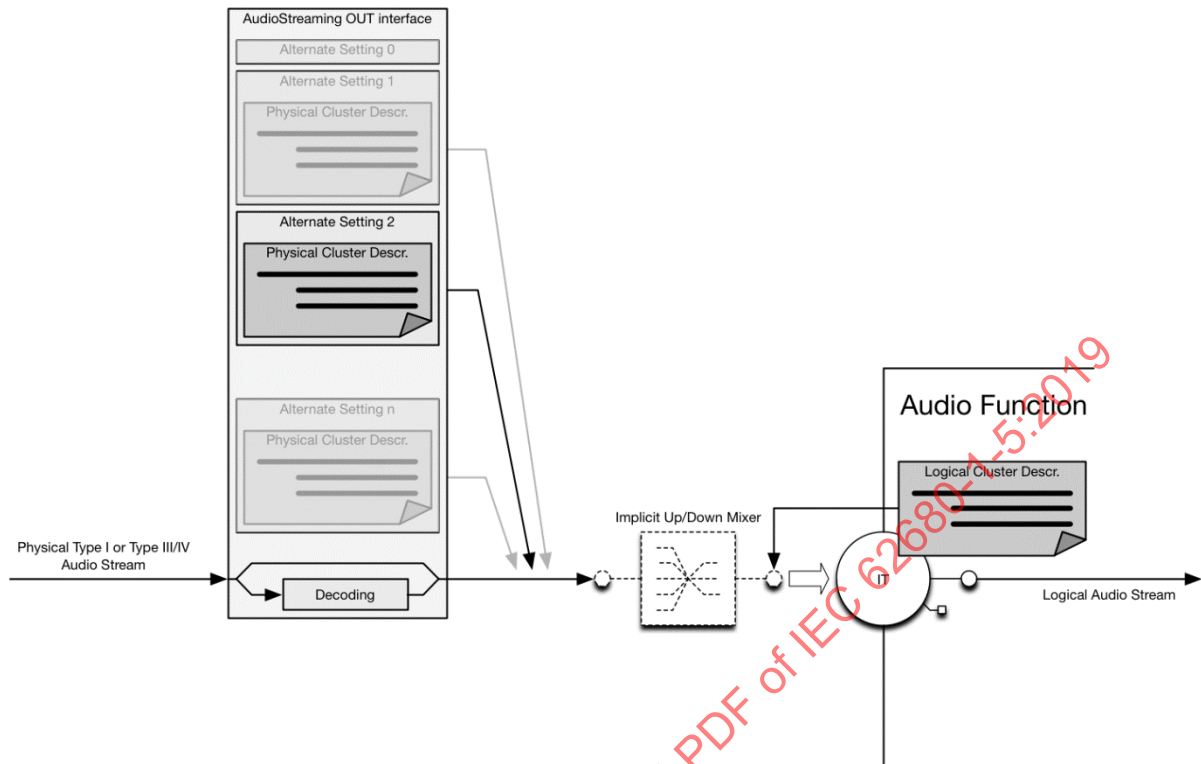
Enfin, cette grappe entre ensuite dans la fonction audio au moyen de la broche de sortie du terminal d'entrée.

Cette construction permet un découplage complet entre la configuration de grappe physique sélectionnée par le paramètre alternatif actif de l'interface et la configuration de grappe logique décrite par le descripteur de grappe du terminal d'entrée. En d'autres termes, la configuration de grappe logique est indépendante du paramètre alternatif choisi pour l'interface AudioStreaming.

Le processus utilisé pour l'opération de mixage élévateur/réducteur est spécifique à la mise en œuvre et n'est pas spécifié ici.

La Figure 4-4 représente le concept d'interfaces AudioStreaming OUT.

Figure 4-4: Mise en correspondance d'une grappe physique vers une grappe logique



Anglais	Français
AudioStreaming Interface OUT	Interface AudioStreaming OUT
Alternate Setting	Paramètre alternatif
Physical Cluster Descr.	Descr. de grappe physique
Decoding	Décodage
Physical Type I or Type III/IV Audio Stream	Flux audio physique de Type I ou de Type III/IV
Implicit Up/Down Mixer	Unité de mixage élévateur/réducteur implicite
Audio Function	Fonction audio
Logical Cluster Descr.	Descr. de grappe logique
Logical Audio Stream	Flux audio logique

4.4.1.2 INTERFACE AUDIOSTREAMING IN – TERMINAL DE SORTIE

Pour les flux audio de Type I, le descripteur de grappe physique référencé par le descripteur d'interface AudioStreaming spécifique à la classe du paramètre alternatif actif de cette interface AudioStreaming OUT décrit les canaux audio qui entrent dans l'interface.

Pour les flux audio de Type III/IV, la grappe physique décrit les canaux audio après leur décodage en canaux audio distincts.

Un descripteur de grappe en amont décrit la grappe logique qui entre dans la broche d'entrée du terminal de sortie.

La grappe entre ensuite dans un processus de mixage éleveur/réducteur implicite, afin de produire une grappe de sortie telle que décrite par le descripteur de grappe physique référencé par le descripteur d'interface AudioStreaming spécifique à la classe du paramètre alternatif actif de l'interface. Les échantillons audio (sous-emplacements audio) doivent être ordonnés exactement de la même façon que dans le descripteur de grappe physique.

Pour les flux audio de Type I, cette grappe quitte ensuite l'interface AudioStreaming.

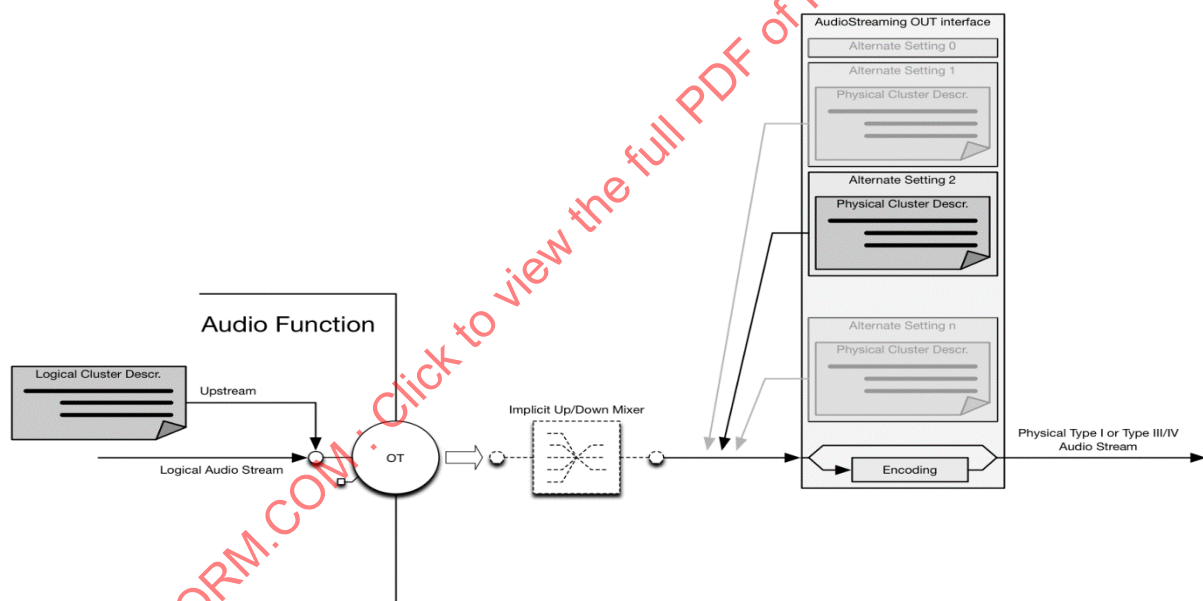
Pour les flux audio de Type III/IV, la grappe est codée avant de quitter l'interface AudioStreaming.

Cette construction permet un découplage complet entre la configuration de grappe logique décrite par le descripteur de grappe en amont et la configuration de grappe physique sélectionnée par le paramètre alternatif actif de l'interface. En d'autres termes, la configuration de grappe physique est indépendante de la configuration de grappe en amont et uniquement déterminée par le paramètre alternatif choisi pour l'interface AudioStreaming.

Le processus utilisé pour l'opération de mixage éleveur/réducteur est spécifique à la mise en œuvre et n'est pas spécifié ici.

La Figure 4-5 représente le concept d'interfaces AudioStreaming IN.

Figure 4-5: Mise en correspondance d'une grappe logique vers une grappe physique



Anglais	Français
Logical Cluster Descr.	Descr. de grappe logique
Audio Function	Fonction audio
Upstream	Amont
Logical Audio Stream	Flux audio logique
Implicit Up/Down Mixer	Unité de mixage éleveur/réducteur implicite
AudioStreaming Interface OUT	Interface AudioStreaming OUT
Alternate Setting	Paramètre alternatif
Physical Cluster Descr.	Descr. de grappe physique
Encoding	Codage
Physical Type I or Type III/IV Audio Stream	Flux audio physique de Type I ou de Type III/IV

4.5 DESCRIPTEURS DE L'INTERFACE AUDIOCONTROL

Les descripteurs d'interface AudioControl (AC) contiennent toutes les informations pertinentes pour caractériser entièrement la fonction audio correspondante. Le descripteur d'interface normalisé caractérise l'interface elle-même, tandis que le descripteur d'interface spécifique à la classe fournit des informations pertinentes concernant les composants internes de la fonction audio. Il spécifie les informations relatives au niveau de révision et répertorie les capacités de chaque unité et terminal.

4.5.1 DESCRIPTEUR D'INTERFACE AC NORMALISE

Le descripteur d'interface AC normalisé est identique au descripteur d'interface normalisé défini dans la *Spécification USB*, sauf que certains champs ont maintenant des valeurs dédiées.

Tableau 4-14: Descripteur d'interface AC normalisé

Décalage	Champ	Taille	Valeur	Description
0	bLength	1	Nombre	Taille de ce descripteur, en octets: 9.
1	bDescriptorType	1	Constante	Type de descripteur INTERFACE.
2	bInterfaceNumber	1	Nombre	Nombre d'interfaces. Valeur à base zéro identifiant l'indice de l'ensemble d'interfaces concurrentes prises en charge par cette configuration.
3	bAlternateSetting	1	Nombre	Valeur utilisée pour choisir ce paramètre alternatif pour l'interface identifiée dans le champ précédent. Doit être défini sur 0.
4	bNumEndpoints	1	Nombre	Nombre de points d'extrémité utilisés par cette interface (excepté le point d'extrémité 0). Ce nombre est 0 ou 1 si le point d'extrémité d'interruption facultatif est présent.
5	bInterfaceClass	1	Classe	AUDIO. Code de classe d'interface audio (attribué par l'USB-IF). Voir A.4, "Code de classe d'interface audio".
6	bInterfaceSubClass	1	Sous-classe	AUDIOCONTROL. Code de sous-classe d'interface audio. Attribué par la présente spécification. Voir A.5, "Codes de sous-classes d'interface audio".
7	bInterfaceProtocol	1	Protocole	Code de protocole d'interface IP_VERSION_03_00. Indique la version actuelle de la spécification. Voir A.6, "Codes de protocoles d'interface audio".
8	iInterface	1	Indice	Indice du descripteur de chaîne décrivant cette interface.

4.5.2 DESCRIPTEUR D'INTERFACE AC SPECIFIQUE A LA CLASSE

Le descripteur d'interface AC spécifique à la classe est une concaténation de tous les descripteurs utilisés pour décrire pleinement la fonction audio, c'est-à-dire tous les descripteurs d'horloge (CD - Clock descriptors), tous les descripteurs d'unité (UD - Unit descriptors), tous les descripteurs de terminal (TD) et tous les descripteurs de domaine d'alimentation (PDD - Power Domain descriptors).

La longueur totale du descripteur d'interface AC spécifique à la classe dépend du nombre d'entités d'horloge, d'unités, de terminaux et de domaines d'alimentation dans la fonction audio. Par conséquent, le descripteur commence par un en-tête qui reflète la longueur totale en octets du descripteur d'interface AC spécifique à la classe dans le champ **wTotalLength**. Le champ **bCategory** contient une constante qui indique l'utilisation principale de cette fonction audio, telle que prévue par le fabricant.

L'ordre dans lequel les descripteurs d'entité d'horloge, d'unité, de terminal et de domaine d'alimentation sont indiqués n'est pas important, car chaque descripteur peut être identifié au moyen de ses champs

bDescriptorType et **bDescriptorSubtype**.

Le tableau suivant définit le descripteur d'en-tête d'interface AC spécifique à la classe.

Tableau 4-15: Descripteur d'en-tête d'interface AC spécifique à la classe

Décalage	Champ	Taille	Valeur	Description
0	bLength	1	Nombre	Taille de ce descripteur, en octets: 10.
1	bDescriptorType	1	Constante	Type de descripteur CS_INTERFACE.
2	bDescriptorSubtype	1	Constante	Sous-type de descripteur HEADER.
3	bCategory	1	Constante	Constante indiquant l'utilisation principale de cette fonction audio, telle que prévue par le fabricant. Voir A.7, "Codes de catégories de fonction audio".
4	wTotalLength	2	Nombre	Nombre total d'octets renvoyés pour le descripteur d'interface AudioControl spécifique à la classe. Inclut la longueur combinée de cet en-tête de descripteur et de tous les descripteurs de source d'horloge, d'unités, de terminaux et de domaines d'alimentation.
6	bmControls	4	Topogramme binaire	D1..0: Commande de latence. D31..2: Réserve.

Les descripteurs d'entité d'horloge, les descripteurs d'unité, les descripteurs de terminal et les descripteurs de domaine d'alimentation apparaissent ensuite, dans n'importe quel ordre. La topologie des descripteurs d'entité dépend du type d'entité d'horloge, d'unité ou de terminal qu'ils représentent. Le descripteur de domaine d'alimentation a une topologie fixe. Il existe trois types de descripteurs d'entité d'horloge, sept types de descripteurs d'unité, deux types de descripteurs de terminal et un seul type de descripteur de domaine d'alimentation. Les descripteurs d'entité sont répertoriés dans les sections suivantes.

Les trois premiers champs sont communs à tous les descripteurs. Ils contiennent la longueur du descripteur, le type de descripteur et le sous-type de descripteur. Les descripteurs d'entité possèdent un champ commun supplémentaire qui contient l'ID d'entité d'horloge, l'ID d'unité, l'ID de terminal ou l'ID de domaine d'alimentation de l'entité.

À chaque entité (horloge, unité, terminal et domaine d'alimentation) au sein de la fonction audio est attribué un numéro d'identification unique, l'ID d'entité d'horloge (CID), l'ID d'unité (UID), l'ID de terminal (TID) ou l'ID de domaine d'alimentation (PID), contenu dans le champ **bClockID**, **bUnitID**, **bTerminalID**, **bPowerDomainID** du descripteur. La valeur 0x00 est réservée pour l'ID non défini, qui limite efficacement le nombre total d'entités adressables dans la fonction audio (entités d'horloge, unités, terminaux et domaines d'alimentation) à 255.

Outre l'identification unique de toutes les entités adressables d'une fonction audio, les ID (à l'exception de l'ID de domaine d'alimentation) servent également à décrire la topologie de la fonction audio; c'est-à-dire que le champ **bSourceID** d'un descripteur d'unité ou de terminal indique à quel autre unité ou terminal cette unité ou ce terminal est connecté. De même, le champ **bCSourceID** d'un descripteur de terminal indique à quelle entité d'horloge ce terminal est connecté. De plus, les ID d'entité sont également utilisés pour indiquer à quel domaine d'alimentation appartient chaque entité.

4.5.2.1 DESCRIPTEUR DE TERMINAL D'ENTREE

Le descripteur de terminal d'entrée (ITD) fournit à l'hôte des informations relatives aux aspects fonctionnels du terminal d'entrée.

Le terminal d'entrée est identifié de manière unique par la valeur du champ **bTerminalID**. Aucun autre terminal ou unité au sein de l'interface AudioControl ne peut avoir le même ID. Cette valeur doit être transmise dans le champ d'identification d'entité (faisant partie du champ **wIndex**) de chaque demande adressée au terminal.

Le champ **wTerminalType** fournit des informations pertinentes sur l'entité physique représentée par le terminal d'entrée. Il peut s'agir d'un point d'extrémité USB OUT, d'une connexion d'entrée de ligne externe, d'un microphone, etc. Une liste complète des codes de type de terminal est fournie dans le document séparé *Types de terminaux audio USB*, qui est vu comme faisant partie de la présente spécification.

Le champ **bAssocTerminal** est utilisé pour associer un terminal de sortie à ce terminal d'entrée, ce qui permet la mise en œuvre efficace d'une paire de terminaux bidirectionnels. Si aucune association n'existe, le champ **bAssocTerminal** doit être défini sur zéro.

Le logiciel hôte peut traiter les terminaux associés comme étant physiquement liés. Dans de nombreux cas, un terminal ne peut pas exister sans l'autre. Un terminal d'entrée, qui représente le microphone, et un terminal de sortie, qui représente l'oreillette d'un micro-casque, constituent un exemple type d'une telle paire de terminaux.

Le champ **bCSourceID** contient une constante indiquant à quelle entité d'horloge la broche d'entrée d'horloge de ce terminal d'entrée est connectée.

Le champ **pwClusterDescrID** contient l'ID unique du descripteur de grappe qui caractérise la grappe logique quittant le terminal d'entrée par la broche de sortie unique (connexion "en aval"). Pour une description détaillée du descripteur de grappe, voir 4.3, "Descripteur de grappe".

Le champ **wExTerminalDescrID** contient l'ID unique du descripteur de terminal étendu associé à ce terminal. Pour une description détaillée du descripteur de terminal étendu, voir 4.5.2.3, "Descripteur de terminal étendu".

Le champ **wConnectorsDescrID** contient l'ID unique du descripteur de connecteurs associé à ce terminal. Pour une description détaillée du descripteur de connecteurs, voir 4.5.2.4, "Descripteur de connecteurs".

L'ID d'un descripteur de chaîne est fourni pour décrire plus précisément le terminal d'entrée.

Le tableau suivant présente les grandes lignes du descripteur de terminal d'entrée.

Tableau 4-16: Descripteur de terminal d'entrée

Décalage	Champ	Taille	Valeur	Description
0	bLength	1	Nombre	Taille de ce descripteur, en octets: 20.
1	bDescriptorType	1	Constante	Type de descripteur CS_INTERFACE.
2	bDescriptorSubtype	1	Constante	Sous-type de descripteur INPUT_TERMINAL.
3	bTerminalID	1	Nombre	Valeur identifiant de manière unique le terminal au sein de la fonction audio. Cette valeur est utilisée dans toutes les demandes d'adressage de ce terminal.
4	wTerminalType	2	Constante	Constante caractérisant le type de terminal. Voir <i>Types de terminaux audio USB</i> .
6	bAssocTerminal	1	Nombre	ID du terminal de sortie auquel ce terminal d'entrée est associé.
7	bCSourceID	1	Nombre	ID de l'entité d'horloge à laquelle ce terminal d'entrée est connecté.
8	bmControls	4	Topogramme binaire	D1..0: Commande d'insertion. D3..2: Commande de surcharge. D5..4: Commande d'engorgement. D7..6: Commande de débordement. D31..8: Réservé.
12	wClusterDescrID	2	Nombre	ID du descripteur de grappe pour ce terminal d'entrée.
14	wExTerminalDescrID	2	Nombre	ID du descripteur de terminal étendu pour ce terminal d'entrée. Doit être défini sur zéro si aucun descripteur de terminal étendu n'est présent.
16	wConnectorsDescrID	2	Nombre	ID du descripteur de connecteurs pour ce terminal d'entrée. Doit être défini sur zéro si aucun descripteur de connecteurs n'est présent.
18	wTerminalDescrStr	2	wStrDescrID	ID d'un descripteur de chaîne spécifique à la classe, décrivant le terminal d'entrée.

4.5.2.2 DESCRIPTEUR DE TERMINAL DE SORTIE

Le descripteur de terminal de sortie (OTD) fournit à l'hôte des informations relatives aux aspects fonctionnels du terminal de sortie.

Le terminal de sortie est identifié de manière unique par la valeur du champ **bTerminalID**. Aucun autre terminal ou unité au sein de l'interface AudioControl ne peut avoir le même ID. Cette valeur doit être transmise dans le champ d'identification d'entité (faisant partie du champ **wIndex**) de chaque demande adressée au terminal.

Le champ **wTerminalType** fournit des informations pertinentes sur l'entité physique représentée par le terminal de sortie. Il peut s'agir d'un point d'extrémité USB IN, d'une connexion de sortie de ligne externe, d'un système de haut-parleurs, etc. Une liste complète des codes de type de terminal est fournie dans le document séparé *Types de terminaux audio USB*, qui est vu comme faisant partie de la présente spécification.

Le champ **bAssocTerminal** est utilisé pour associer un terminal d'entrée à ce terminal de sortie, ce qui permet la mise en œuvre efficace d'une paire de terminaux bidirectionnels. Si aucune association n'existe, le champ **bAssocTerminal** doit être défini sur zéro.

Le logiciel hôte peut traiter les terminaux associés comme étant physiquement liés. Dans de nombreux cas, un terminal ne peut pas exister sans l'autre. Un terminal d'entrée, qui représente le microphone, et un terminal de sortie, qui représente l'oreillette d'un micro-casque, constituent un exemple type d'une telle paire de terminaux.

Le champ **bSourceID** est utilisé pour décrire la connectivité de ce terminal. Il contient l'ID de l'unité ou du terminal auquel ce terminal de sortie est connecté au moyen de sa broche d'entrée. Le descripteur de grappe, qui décrit les canaux logiques entrant dans le terminal de sortie, n'est pas répété ici. Il appartient au logiciel hôte de tracer la connexion "en amont" pour localiser le descripteur de grappe relatif à cette grappe.

Le champ **bCSourceID** contient une constante indiquant à quelle entité d'horloge la broche d'entrée d'horloge de ce terminal de sortie est connectée.

Le champ **wExTerminalDescrID** contient l'ID unique du descripteur de terminal étendu associé à ce terminal. Pour une description détaillée du descripteur de terminal étendu, voir 4.5.2.3, "Descripteur de terminal étendu".

Le champ **wConnectorsDescrID** contient l'ID unique du descripteur de connecteurs associé à ce terminal. Pour une description détaillée du descripteur de connecteurs, voir 4.5.2.4, "Descripteur de connecteurs".

L'ID d'un descripteur de chaîne est fourni pour décrire plus précisément le terminal de sortie.

Le tableau suivant présente les grandes lignes du descripteur de terminal de sortie.

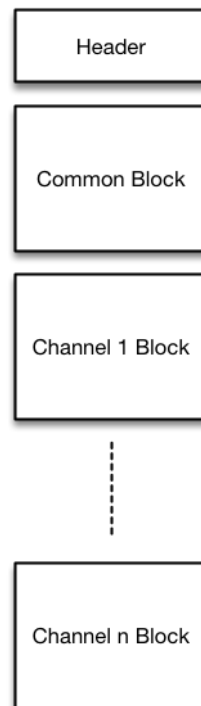
Tableau 4-17: Descripteur de terminal de sortie

Décalage	Champ	Taille	Valeur	Description
0	bLength	1	Nombre	Taille de ce descripteur, en octets: 19.
1	bDescriptorType	1	Constante	Type de descripteur CS_INTERFACE.
2	bDescriptorSubtype	1	Constante	Sous-type descripteur OUTPUT_TERMINAL.
3	bTerminalID	1	Nombre	Valeur identifiant de manière unique le terminal au sein de la fonction audio. Cette valeur est utilisée dans toutes les demandes d'adressage de ce terminal.
4	wTerminalType	2	Constante	Constante caractérisant le type de terminal. Voir <i>Types de terminaux audio USB</i> .
6	bAssocTerminal	1	Nombre	Valeur identifiant le terminal d'entrée auquel ce terminal de sortie est associé.
7	bSourceID	1	Nombre	ID de l'unité ou du terminal auquel ce terminal est connecté.
8	bCSourceID	1	Nombre	ID de l'entité d'horloge à laquelle ce terminal de sortie est connecté.
9	bmControls	4	Topogramme binaire	D1..0: Commande d'insertion. D3..2: Commande de surcharge. D5..4: Commande d'engorgement. D7..6: Commande de débordement. D31..8: Réserve.
13	wExTerminalDescrID	2	Nombre	ID du descripteur de terminal étendu pour ce terminal de sortie. Doit être défini sur zéro si aucun descripteur de terminal étendu n'est présent.
15	wConnectorsDescrID	2	Nombre	ID du descripteur de connecteurs pour ce terminal de sortie. Doit être défini sur zéro si aucun descripteur de connecteurs n'est présent.
17	wTerminalDescrStr	2	wStrDescrID	ID d'un descripteur de chaîne spécifique à la classe, décrivant le terminal de sortie.

4.5.2.3 DESCRIPTEUR DE TERMINAL ETENDU

Le descripteur de terminal étendu renvoie en option des informations physiques supplémentaires concernant les canaux qui entrent ou sortent du terminal. Il utilise toujours la représentation haute capacité.

Figure 4-6: Descripteur de terminal étendu



Anglais	Français
Header	En-tête
Common Block	Bloc commun
Channel 1 Block	Bloc canal 1
Channel n Block	Bloc canal n

Le descripteur est constitué d'un en-tête fixe, suivi d'un bloc commun facultatif, suivi d'autant de blocs de canal qu'il y a de canaux dans la grappe du terminal.

4.5.2.3.1 EN-TÊTE DU DESCRIPTEUR DE TERMINAL ÉTENDU

L'en-tête commence par le champ **wLength** qui contient le nombre total d'octets dans tout le descripteur.

Le champ **wDescriptorID** contient un numéro ID qui identifie de manière unique le descripteur au sein de la fonction audio. La valeur zéro est réservée et ne doit pas être utilisée comme ID de descripteur de terminal étendu valide.

Le champ **bNrChannels** indique le nombre de canaux audio présents dans la grappe logique du terminal.

Tableau 4-18: En-tête du descripteur de terminal étendu

Décalage	Champ	Taille	Valeur	Description
0	wLength	2	Nombre	Longueur totale du descripteur de grappe, en octets.
2	bDescriptorType	1	Constante	Type de descripteur CS_INTERFACE.
3	bDescriptorSubtype	1	Constante	Sous-type de descripteur EXTENDED_TERMINAL.
4	wDescriptorID	2	Nombre	ID unique de ce descripteur de terminal étendu.
6	bNrChannels	1	Nombre	Nombre de canaux présents dans la grappe du terminal: n.

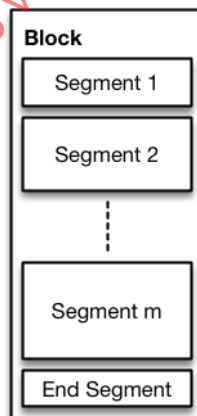
4.5.2.3.2 BLOC DE DESCRIPTEUR DE TERMINAL ETENDU

L'en-tête du descripteur de terminal étendu est suivi d'un certain nombre de blocs de descripteur de terminal étendu. Il y a un bloc commun facultatif, suivi d'autant de blocs de canal qu'il y a de canaux dans la grappe du terminal. Chaque bloc est composé d'un certain nombre de segments.

Le bloc commun est composé de segments qui contiennent des informations pertinentes sur les caractéristiques du terminal dans son ensemble.

Un bloc de canal est composé de segments qui contiennent des informations pertinentes sur les caractéristiques physiques du canal. Tous les segments sont facultatifs. Il est fortement recommandé d'utiliser la même topologie pour chaque bloc de canal, c'est-à-dire que les mêmes segments apparaissent dans le même ordre dans chaque bloc de canal. La Figure 4-7 représente plus en détail les concepts ci-dessus.

Figure 4-7: Bloc de canal de terminal étendu



Anglais	Français
Block	Bloc
End Segment	Segment de fin

4.5.2.3.3 SEGMENTS

Il existe deux types de segments: Les segments de bloc commun contiennent des informations pertinentes sur le terminal dans son ensemble. Les segments de bloc de canal contiennent des informations pertinentes sur certains aspects d'un canal particulier dans la grappe du terminal. Les deux types de segments partagent la même topologie.

Le champ **bSegmentType** décrit le type de segment (bloc commun ou bloc de canal) ainsi que le type de contenu dans le segment.

La topologie d'un segment est toujours la suivante:

Tableau 4-19: Segment de descripteur de grappe

Décalage	Champ	Taille	Valeur	Description
0	wLength	2	Nombre	Longueur du segment, en octets: 3+n.
2	bSegmentType	1	Constante	Décrit le type de segment et le type de contenu du segment.
3	Spécifique au segment	n		Contenu spécifique au segment.

4.5.2.3.3.1 SEGMENT DE FIN

Chaque bloc se termine par un segment de fin. Le segment de fin marque la fin du bloc à longueur variable. Le segment de fin ne comporte pas de section spécifique au segment et est structuré comme suit:

Tableau 4-20: Segment de fin

Décalage	Champ	Taille	Valeur	Description
0	wLength	2	Nombre	Longueur du segment de fin, en octets: 3.
2	bSegmentType	1	Constante	END_SEGMENT.

4.5.2.3.3.2 SEGMENTS DE BLOCS COMMUNS

Les types de segments de blocs communs suivants sont définis:

- Spécifique au fournisseur

Les valeurs des types de segments de bloc commun peuvent être consultées en A.18, "Types de segments de terminal étendus".

4.5.2.3.3.2.1 SEGMENT DÉFINI PAR LE FOURNISSEUR

Les fournisseurs sont autorisés à ajouter des segments définis par le fournisseur au bloc commun afin de transmettre des informations propriétaires supplémentaires sur le terminal. Le segment défini par le fournisseur doit utiliser la topologie suivante:

Tableau 4-21: Segment défini par le fournisseur

Décalage	Champ	Taille	Valeur	Description
0	wLength	2	Nombre	Longueur du segment de fin, en octets: 3+n.
2	bSegmentType	1	Constante	TERMINAL_VENDOR_DEFINED.
3	Défini par le fournisseur	n	N/A	Extension du segment définie par le fournisseur.

4.5.2.3.3.3 SEGMENTS DE BLOCS DE CANAUX

Les types de segments de blocs de canaux suivants sont définis:

- Bande passante
- Réponse en amplitude
- Réponse en amplitude/phase
- Position
- Spécifique au fournisseur

Les valeurs des types de segments de blocs de canaux peuvent être consultées à l'Annexe A.18, "Types de segments de terminal étendus".

4.5.2.3.3.3.1 SEGMENT DE BANDE PASSANTE

Le segment de bande passante contient des informations de base sur la bande passante audio du canal. La bande passante est spécifiée par les points de fréquence -3 dB inférieurs et supérieurs, en Hz, de la bande disponible dans les champs **dMinFreq** et **dMaxFreq**.

Tableau 4-22: Segment de bande passante

Décalage	Champ	Taille	Valeur	Description
0	wLength	2	Nombre	Longueur du segment de bande passante, en octets: 11.
2	bSegmentType	1	Constante	CHANNEL_BANDWIDTH.
3	dMinFreq	4	Nombre	Point de fréquence -3 dB inférieur.
7	dMaxFreq	4	Nombre	Point de fréquence -3 dB supérieur.

4.5.2.3.3.3.2 REPONSE EN AMPLITUDE

Le segment de réponse en amplitude contient des informations détaillées concernant l'amplitude de la fonction de transfert (réponse en fréquence) du canal. L'amplitude est spécifiée sous la forme d'une matrice de paires [point de fréquence, amplitude]. Les valeurs de fréquence sont spécifiées en Hz et les valeurs d'amplitude peuvent aller de +127,9961 dB (0x7FFF) à -127,9961 dB (0x8001) par paliers de 1/256 dB ou de 0,00390625 dB (0x0001). De plus, le code 0x8000, représentant le silence (c'est à dire $-\infty$ dB), peut également être utilisé.

Tableau 4-23: Segment d'amplitude

Décalage	Champ	Taille	Valeur	Description
0	wLength	2	Nombre	Longueur du segment de réponse en amplitude, en octets: $3+n*6$.
2	bSegmentType	1	Constante	CHANNEL_MAGNITUDE_RESPONSE.
3	dFreq(1)	4	Nombre	Premier point de fréquence.
7	wMagnitude(1)	2	Nombre	Première valeur d'amplitude.
...	...	...	...	
$3+(n-1)*6$	dFreq(n)	4	Nombre	Dernier point de fréquence.
$7+(n-1)*6$	wMagnitude(n)	2	Nombre	Dernière valeur d'amplitude.

4.5.2.3.3.3.3 REPONSE EN AMPLITUDE/PHASE

Le segment de réponse en amplitude/phase contient des informations détaillées concernant l'amplitude et la phase de la fonction de transfert (réponse en fréquence) du canal. L'amplitude/phase est spécifiée sous la forme d'une matrice de triplets [point de fréquence, amplitude, phase]. Les valeurs de fréquence sont spécifiées en Hz. Les valeurs d'amplitude peuvent aller de +127,9961 dB (0x7FFF) à -127,9961 dB (0x8001) par paliers de 1/256 dB ou de 0,00390625 dB (0x0001). De plus, le code 0x8000, représentant le silence (c'est à dire $-\infty$ dB), peut également être utilisé. Les valeurs de phase peuvent aller de $+0,99996948242 * \pi$ à $-\pi$ (0x8000) par paliers de $1/32768 * \pi$ (0x0001).

Tableau 4-24: Segment d'amplitude/phase

Décalage	Champ	Taille	Valeur	Description
0	wLength	2	Nombre	Longueur du segment de réponse en amplitude, en octets: $3+n*8$.
2	bSegmentType	1	Constante	CHANNEL_MAGNITUDE/PHASE_RESPONSE.
3	dFreq(1)	4	Nombre	Premier point de fréquence.
7	wMagnitude(1)	2	Nombre	Première valeur d'amplitude.
9	wPhase(1)	2	Nombre	Première valeur de phase.
...	...	...	...	...
$3+(n-1)*8$	dFreq(n)	4	Nombre	Dernier point de fréquence.
$7+(n-1)*8$	wMagnitude(n)	2	Nombre	Dernière valeur d'amplitude.
$9+(n-1)*8$	wPhase(n)	2	Nombre	Dernière valeur de phase.

4.5.2.3.3.3.4 SEGMENT POSITION_XYZ

Le segment Position_XYZ contient les coordonnées cartésiennes (X, Y, Z) de la source ou du destinataire associé au canal. Les valeurs X, Y et Z sont exprimées en micromètres (μm) et sont relatives à une origine non spécifiée à (0, 0, 0). La fonction audio peut disposer de moyens hors bande pour indiquer à l'hôte où est située l'origine réelle du système de coordonnées sur le dispositif. La valeur 0xFFFFFFFF a une signification particulière. Elle est utilisée pour indiquer que la coordonnée d'une dimension donnée est inconnue.

Tableau 4-25: Segment de position

Décalage	Champ	Taille	Valeur	Description
0	wLength	2	Nombre	Longueur du segment, en octets: 15.
2	bSegmentType	1	Constante	CHANNEL_POSITION_XYZ.
3	dX	4	Nombre	Coordonnée X de la source ou du destinataire audio associé au canal.
7	dY	4	Nombre	Coordonnée Y de la source ou du destinataire audio associé au canal.
11	dZ	4	Nombre	Coordonnée Z de la source ou du destinataire audio associé au canal.

4.5.2.3.3.3.5 SEGMENT POSITION_RΘΦ

Le segment Position_RΘΦ contient les coordonnées sphériques (R, Θ, Φ) de la source ou du destinataire associé au canal. Les valeurs R, Θ, et Φ sont respectivement exprimées en μm, μrad et μrad, et sont relatives à une origine non spécifiée à (0, 0, 0). La fonction audio peut disposer de moyens hors bande pour indiquer à l'hôte où est située l'origine réelle du système de coordonnées sur le dispositif. La valeur 0xFFFFFFFF a une signification particulière. Elle est utilisée pour indiquer que la coordonnée d'une dimension donnée est inconnue.

Tableau 4-26: Segment de position

Décalage	Champ	Taille	Valeur	Description
0	wLength	2	Nombre	Longueur du segment, en octets: 15.
2	bSegmentType	1	Constante	CHANNEL_POSITION_RΘΦ.
3	dR	4	Nombre	Coordonnée R de la source ou du destinataire audio associé au canal.
7	dΘ	4	Nombre	Coordonnée Θ de la source ou du destinataire audio associé au canal.
11	dΦ	4	Nombre	Coordonnée Φ de la source ou du destinataire audio associé au canal.

4.5.2.3.3.3.6 SEGMENT DEFINI PAR LE FOURNISSEUR

Les fournisseurs sont autorisés à ajouter des segments définis par le fournisseur au bloc de canal afin de transmettre des informations propriétaires supplémentaires sur le canal. Le segment défini par le fournisseur doit utiliser la topologie suivante:

Tableau 4-27: Segment défini par le fournisseur

Décalage	Champ	Taille	Valeur	Description
0	wLength	2	Nombre	Longueur du segment de fin, en octets: 3+n.
2	bSegmentType	1	Constante	CHANNEL_VENDOR_DEFINED.
3	Défini par le fournisseur	n	N/A	Extension du segment définie par le fournisseur.

4.5.2.4 DESCRIPTEUR DE CONNECTEURS

Le descripteur de connecteurs facultatif renvoie des informations concernant les attributs physiques de tous les connecteurs associés à un terminal d'entrée ou de sortie. Le descripteur de connecteurs utilise toujours la représentation haute capacité.

Le champ **wDescriptorID** contient un numéro ID qui identifie de manière unique le descripteur au sein de la fonction audio. La valeur zéro est réservée et ne doit pas être utilisée comme ID de descripteur de connecteurs valide.

Le champ **bNrConnectors** indique le nombre de connecteurs physiques distincts associés à ce terminal. Les connecteurs sont numérotés à partir de 1 jusqu'au nombre total de connecteurs associés à ce terminal. Pour chaque connecteur, le descripteur contient un groupe de champs qui décrivent plus précisément le connecteur.

Le champ **baConID(i)** contient un identificateur unique pour Connector(i). L'utilisation principale sert à indiquer que le même connecteur est associé à plusieurs terminaux. Par exemple, un connecteur de micro-casque intègre les signaux du casque stéréo du micro-casque ainsi que le signal du microphone mono. Ce connecteur fait par conséquent partie du terminal de sortie qui représente le casque stéréo et fait également partie du terminal d'entrée qui représente le microphone. Ce connecteur est ensuite répertorié à la fois dans le descripteur de connecteurs du terminal d'entrée et dans le descripteur de connecteurs du terminal de sortie, en utilisant la même valeur de **baConID(i)** dans les deux descripteurs pour indiquer la liaison. Noter que la valeur de **baConID(i)** n'est jamais utilisée comme identificateur d'une entité adressable et que, par conséquent, les valeurs de **baConID(i)** peuvent chevaucher les valeurs d'ID d'entité.

Le **waClusterDescrID(i)** contient l'ID unique d'un descripteur de grappe physique, décrivant les canaux physiques réels transportés sur Connector(i). Noter que, dans la plupart des cas, l'union de tous les descripteurs de grappe physique, référencés par le descripteur de connecteurs, est identique au descripteur de grappe logique, associé à la broche du terminal. Il est admis que les canaux apparaissent dans plus d'un descripteur de grappe physique pour répondre aux situations où les mêmes canaux physiques sont disponibles sur différents connecteurs. Par exemple, le même canal peut être disponible sur un connecteur XLR symétrique ainsi que sur un connecteur BNC asymétrique. Pour une description détaillée du descripteur de grappe, voir 4.3, "Descripteur de grappe".

Le champ **baConType(i)** contient une valeur qui identifie l'apparence physique de Connector(i). Les définitions des constantes du champ **baConType(i)** peuvent être consultées en A.24, "Types de connecteurs".

Le champ **bmaConAttributes(i)** contient un topogramme binaire qui identifie le genre de Connector(i) (D1..0) et indique également si cette commande est capable de détecter l'insertion et le retrait. Si ce bit est défini pour Connector(i), alors le bit D_{i-1} du topogramme binaire renvoyé par la demande de commande d'insertion indique si ce connecteur est inséré ($D_{i-1} = 1$) ou non ($D_{i-1} = 0$). Si ce bit n'est pas défini pour Connector(i), alors le bit D_{i-1} du topogramme binaire renvoyé par la demande de commande d'insertion n'a aucune signification et doit être ignoré.

Le champ **waConDescrStr(i)** est l'ID d'un descripteur de chaîne qui contient un identificateur lisible par l'humain pour Connector(i). Il est recommandé que cette chaîne facilite l'identification unique du connecteur sur le châssis du dispositif. Cela peut être obtenu en étiquetant explicitement le connecteur sur l'enveloppe du dispositif, avec le contenu de la chaîne **waConDescrStr(i)**.

Le champ **daConColor(i)** contient soit 0x00 dans l'octet supérieur et la couleur codée RVB de Connector(i) dans les 3 octets inférieurs, soit 0x01 dans l'octet supérieur et 0x000000 dans les 3 octets inférieurs pour indiquer une couleur non spécifiée.

Tableau 4-28: Descripteur de connecteurs

Décalage	Champ	Taille	Valeur	Description
0	wLength	2	Nombre	Taille de ce descripteur, en octets: 7+n*11.
2	bDescriptorType	1	Constante	Type de descripteur CS_INTERFACE.
3	bDescriptorSubtype	1	Constante	Sous-type de descripteur CONNECTORS.
4	wDescriptorID	2	Nombre	ID unique de ce descripteur de connecteurs.
6	bNrConnectors	1	Nombre	Nombre de connecteurs associés à ce terminal: n
7	baConID(1)	1	Nombre	ID de Connector(1). Peut être utilisé pour indiquer que le même connecteur est associé à plusieurs terminaux.
8	waClusterDescrID(1)	2	Nombre	ID du descripteur de grappe physique transporté sur Connector(1).
10	baConType(1)	1	Constante	Type de connecteur de Connector(1).
11	bmaConAttributes(1)	1	Topogramme binaire	D1..0: Genre: 00: Neutre. 01: Mâle. 10: Femelle. 11: Réservé. D2: Signalement de branchement/débranchement: 0: Non. 1: Oui. D7..3: Réservé.
12	waConDescrStr(1)	2	wStrDescrID	ID d'un descripteur de chaîne qui fournit une description physique de Connector(1).
14	daConColor(1)	4	Nombre	Couleur de connecteur de Connector(1).
...	...	...	...	...
7+(n-1)*11	baConID(n)			Identificateur unique de Connector(n). Peut être utilisé pour indiquer que le même connecteur est associé à plusieurs terminaux.
8+(n-1)*11	waClusterDescrID(n)	2	Nombre	ID unique du descripteur de grappe physique transporté sur Connector(n).
10+(n-1)*11	baConType(n)	1	Constante	Type de connecteur de Connector(n).
11+(n-1)*11	bmaConAttributes(n)	1	Topogramme binaire	D1..0: Genre: 00: Neutre. 01: Mâle. 10: Femelle. 11: Réservé. D2: Signalement de branchement/débranchement: 0: Non. 1: Oui. D7..3: Réservé.

Décalage	Champ	Taille	Valeur	Description
12+(n-1)*11	waConDescrStr(n)	2	wStrDescrID	ID d'un descripteur de chaîne qui fournit une description physique de Connector(n).
14+(n-1)*11	daConColor(n)	4	Nombre	Couleur de connecteur de Connector(n).

4.5.2.5 DESCRIPTEUR D'UNITE DE MIXAGE

L'unité de mixage est identifiée de manière unique par la valeur du champ **bUnitID** du descripteur d'unité de mixage (MUD). Aucun autre terminal ou unité au sein de l'interface AudioControl ne peut avoir le même ID. Cette valeur doit être transmise dans le champ d'identification d'entité (faisant partie du champ **wIndex**) de chaque demande adressée à l'unité de mixage.

Le champ **bNrInPins** contient le nombre de broches d'entrée (p) de l'unité de mixage. Cela équivaut évidemment au nombre de grappes entrant dans l'unité de mixage. La connectivité des broches d'entrée est décrite au moyen de la matrice **baSourceID()**, contenant p éléments. L'indice i de la matrice est à base un et est directement lié aux numéros des broches d'entrée. **baSourceID(i)** contient l'ID de l'unité ou du terminal auquel la broche d'entrée i est connectée. Les descripteurs de grappe, qui décrivent les canaux logiques entrant dans l'unité de mixage, ne sont pas répétés ici. Il appartient au logiciel hôte de tracer les connexions "en amont" afin de localiser les descripteurs de grappe relatifs aux grappes.

Comme une unité de mixage peut redéfinir les localisations spatiales des canaux de sortie logiques contenus dans sa grappe de sortie, un descripteur de grappe de sortie de mixage est nécessaire.

Le **wClusterDescrID** contient l'ID unique du descripteur de grappe qui caractérise la grappe quittant l'unité de mixage par la broche de sortie unique (connexion "en aval"). Pour une description détaillée du descripteur de grappe, voir 4.3, "Descripteur de grappe".

Comme indiqué précédemment, chaque canal d'entrée peut être virtuellement combiné à tous les canaux de sortie. Si n est le nombre total de canaux d'entrée logiques, contenus dans toutes les grappes qui entrent dans l'unité de mixage:

$$n = \sum_{i=1}^{\text{nombre de grappes}} (\text{nombre de canaux logiques dans la grappe } i)$$

et m est le nombre de canaux de sortie logiques, alors il y a $(n \cdot m)$ commandes de mixage dans l'unité de mixage, certaines pouvant ne pas être programmables.

Note: $(n \cdot m)$ doit être limité à 256.

Le descripteur d'unité de mixage indique les commandes programmables dans le champ de topogramme binaire **bmMixerControls**. Ce topogramme binaire doit être interprété comme une matrice de bits bidimensionnelle comportant une ligne pour chaque canal d'entrée logique et une colonne pour chaque canal de sortie logique. Si un bit à la position $[u, v]$ est défini sur un, cela signifie que l'unité de mixage contient une commande de mixage programmable qui connecte le canal d'entrée u au canal de sortie v . Si le bit $[u, v]$ est défini sur zéro, cela indique que la connexion entre le canal d'entrée u et le canal de sortie v n'est pas programmable. La plage valide pour u est comprise entre un et n . La plage valide pour v est comprise entre un et m .

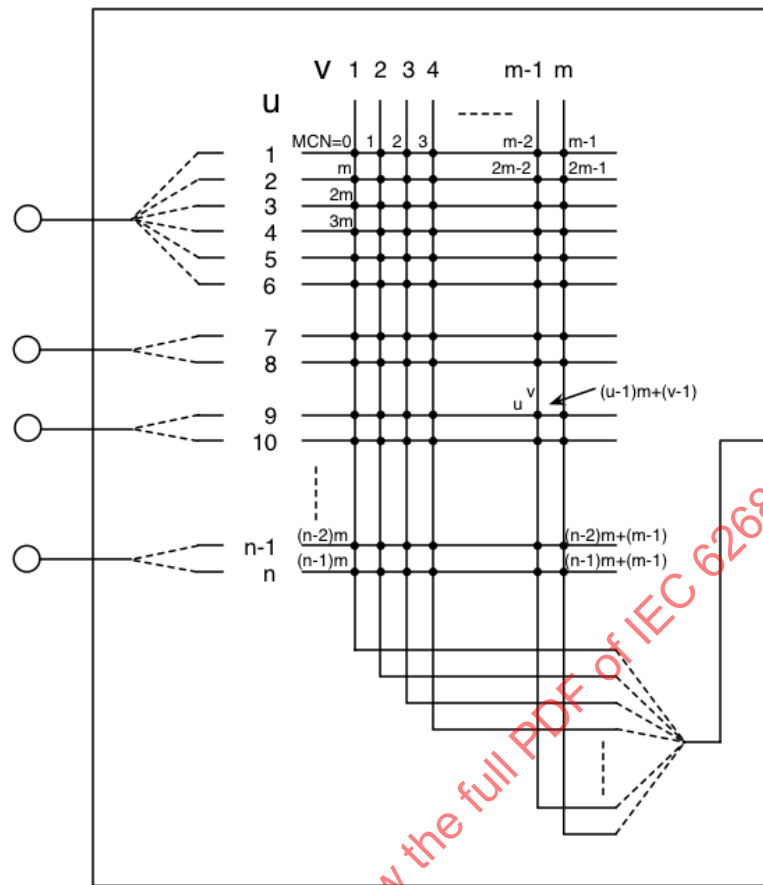
A chaque commande de mixage est attribué un numéro de commande de mixage (MCN - Mixer Control Number) unique. Ce numéro est utilisé pour adresser une commande de mixage particulière dans une demande de commande de mixage "Get"/"Set". Le MCN est calculé comme suit:

$$MCN = (u - 1) \cdot m + (v - 1)$$

La figure suivante représente une explication plus graphique.

Copyright © 1997-2016 USB Implementers Forum, Inc. Tous droits réservés.

Figure 4-8: Composantes internes de l'unité de mixage



Le paramètre actuel de la commande de mixage (programmable et fixe) à n'importe quelle position de la matrice peut toujours être récupéré à l'aide de la demande appropriée. Par conséquent, l'unité de mixage doit toujours mettre en œuvre la demande "Get" avec l'attribut CUR pour chaque nœud de la matrice. Voir 5.2.1.7, "Demande de commande d'unité de mixage" pour plus d'informations.

Le champ **bmMixerControls** stocke la matrice de bits ligne par ligne, le MSb du premier octet (octet supérieur) correspondant à la connexion entre le canal d'entrée 1 et le canal de sortie 1. Si $(n \cdot m)$ n'est pas un nombre entier multiple de 8, la matrice de bits est complétée par des zéros de fin jusqu'à ce qu'un nombre entier d'octets soit occupé. Le nombre d'octets utilisés pour stocker la matrice de bits, N , peut être calculé comme suit:

SI $((n \cdot m) \text{ MOD } 8) \neq 0$ ALORS

$$N = ((n \cdot m) \text{ DIV } 8) + 1$$

SINON

$$N = ((n \cdot m) \text{ DIV } 8)$$

L'ID d'un descripteur de chaîne est fourni pour décrire plus précisément l'unité de mixage.

Le tableau suivant décrit la structure du descripteur d'unité de mixage.

Tableau 4-29: Descripteur d'unité de mixage

Décalage	Champ	Taille	Valeur	Description
0	bLength	1	Nombre	Taille de ce descripteur, en octets: 13+p+N.
1	bDescriptorType	1	Constante	Type de descripteur CS_INTERFACE.
2	bDescriptorSubtype	1	Constante	Sous-type de descripteur MIXER_UNIT.
3	bUnitID	1	Nombre	Valeur identifiant de manière unique l'unité au sein de la fonction audio. Cette valeur est utilisée dans toutes les demandes d'adressage de cette unité.
4	bNrInPins	1	Nombre	Nombre de broches d'entrée de cette unité: p
5	baSourceID(1)	1	Nombre	ID de l'unité ou du terminal auquel la première broche d'entrée de cette unité de mixage est connectée.
...	...	...	...	...
5+(p-1)	baSourceID (p)	1	Nombre	ID de l'unité ou du terminal auquel la dernière broche d'entrée de cette unité de mixage est connectée.
5+p	wClusterDescrID	2	Nombre	ID du descripteur de grappe pour cette unité de mixage.
7+p	bmMixerControls	N	Nombre	Topogramme binaire indiquant quelles commandes de mixage sont programmables.
7+p+N	bmControls	4	Topogramme binaire	D1..0: Commande d'engorgement. D3..2: Commande de débordement. D31..4: Réservé.
11+p+N	wMixerDescrStr	2	wStrDescrID	ID d'un descripteur de chaîne spécifique à la classe, décrivant l'unité de mixage.

4.5.2.6 DESCRIPTEUR D'UNITÉ DE SÉLECTION

L'unité de sélection est identifiée de manière unique par la valeur du champ **bUnitID** du descripteur d'unité de sélection (SUD). Aucun autre terminal ou unité au sein de l'interface AudioControl ne peut avoir le même ID. Cette valeur doit être transmise dans le champ d'identification d'entité (faisant partie du champ **wIndex**) de chaque demande adressée à l'unité de sélection.

Le champ **bNrInPins** contient le nombre de broches d'entrée (*p*) de l'unité de sélection. La connectivité des broches d'entrée est décrite au moyen de la matrice **baSourceID()**, qui contient *p* éléments. L'indice *i* de la matrice est à base un et est directement lié aux numéros des broches d'entrée. **baSourceID(i)** contient l'ID de l'unité ou du terminal auquel la broche d'entrée *i* est connectée.

Les descripteurs de grappe, qui décrivent les canaux logiques entrant dans l'unité de sélection, ne sont pas répétés ici. Pour qu'une unité de sélection soit légalement connectée, *toutes* les grappes qui entrent dans l'unité de sélection doivent présenter exactement le même contenu de descripteur de grappe.

L'ID d'un descripteur de chaîne est fourni pour décrire plus précisément l'unité de sélection.

Le tableau suivant décrit la structure du descripteur d'unité de sélection.

Tableau 4-30: Descripteur d'unité de sélection

Décalage	Champ	Taille	Valeur	Description
0	bLength	1	Nombre	Taille de ce descripteur, en octets: 11+p.
1	bDescriptorType	1	Constante	Type de descripteur CS_INTERFACE.
2	bDescriptorSubtype	1	Constante	Sous-type de descripteur SELECTOR_UNIT.
3	bUnitID	1	Nombre	Valeur identifiant de manière unique l'unité au sein de la fonction audio. Cette valeur est utilisée dans toutes les demandes d'adressage de cette unité.
4	bNrInPins	1	Nombre	Nombre de broches d'entrée de cette unité: p
5	baSourceID(1)	1	Nombre	ID de l'unité ou du terminal auquel la première broche d'entrée de cette unité de sélection est connectée.
...	...	...	...	...
5+(p-1)	baSourceID (p)	1	Nombre	ID de l'unité ou du terminal auquel la dernière broche d'entrée de cette unité de sélection est connectée.
5+p	bmControls	4	Topogramme binaire	D1..0: Commande de sélection. D31..2: Réserve.
9+p	wSelectorDescrStr	2	wStrDescrID	ID d'un descripteur de chaîne spécifique à la classe, décrivant l'unité de sélection.

4.5.2.7 DESCRIPTEUR D'UNITE DE FONCTION

L'unité de fonction est identifiée de manière unique par la valeur du champ **bUnitID** du descripteur d'unité de fonction (FUD). Aucun autre terminal ou unité au sein de l'interface AudioControl ne peut avoir le même ID. Cette valeur doit être transmise dans le champ d'identification d'entité (faisant partie du champ **wIndex**) de chaque demande adressée à l'unité de fonction.

Le champ **bSourceID** est utilisé pour décrire la connectivité de cette unité de fonction. Il contient l'ID de l'unité ou du terminal auquel cette unité de fonction est connectée au moyen de sa broche d'entrée. Le descripteur de grappe, qui décrit les canaux logiques entrant dans l'unité de fonction, n'est pas répété ici. Il appartient au logiciel hôte de tracer la connexion "en amont" pour localiser le descripteur de grappe relatif à cette grappe.

L'ID d'un descripteur de chaîne est fourni pour décrire plus précisément l'unité de fonction.

La topologie du descripteur d'unité de fonction est décrite dans le tableau suivant.

Tableau 4-31: Descripteur d'unité de fonction

Décalage	Champ	Taille	Valeur	Description
0	bLength	1	Nombre	Taille de ce descripteur, en octets: $7+(ch+1)*4$
1	bDescriptorType	1	Constante	Type de descripteur CS_INTERFACE.
2	bDescriptorSubtype	1	Constante	Sous-type de descripteur FEATURE_UNIT.
3	bUnitID	1	Nombre	Valeur identifiant de manière unique l'unité au sein de la fonction audio. Cette valeur est utilisée dans toutes les demandes d'adressage de cette unité.
4	bSourceID	1	Nombre	ID de l'unité ou du terminal auquel cette unité de fonction est connectée.
5	bmaControls(0)	4	Topogramme binaire	Topogramme binaire des commandes du canal maître 0: D1..0: Commande d'inhibition. D3..2: Commande de volume. D5..4: Commande des graves. D7..6: Commande des médiums. D9..8: Commande des aigus. D11..10: Commande de l'égaliseur graphique. D13..12: Commande automatique de gain. D15..14: Commande de retard. D17..16: Commande d'amplification des basses. D19..18: Commande d'intensité sonore. D21..20: Commande de gain d'entrée. D23..22: Commande d'atténuateur de gain d'entrée. D25..24: Commande d'inverseur de phase. D27..26: Commande d'engorgement. D29..28: Commande de débordement. D31..30: Réservé.
$5+(1*4)$	bmaControls(1)	4	Topogramme binaire	Topogramme binaire du canal logique 1.
...	...	...	...	...
$5+(ch*4)$	bmaControls(ch)	4	Topogramme binaire	Topogramme binaire des commandes du canal logique ch.
$5+(ch+1)*4$	wFeatureDescrStr	2	wStrDescrID	ID d'un descripteur de chaîne spécifique à la classe, décrivant cette unité de fonction.

4.5.2.8 DESCRIPTEUR D'UNITE DE CONVERSION DU DEBIT D'ECHANTILLONNAGE

Le descripteur d'unité de conversion du débit d'échantillonnage (RUD) fournit à l'hôte des informations relatives aux aspects fonctionnels de l'unité SRC.

L'unité SRC est identifiée de manière unique par la valeur du champ **bUnitID**. Aucun autre terminal ou unité au sein de l'interface AudioControl ne peut avoir le même ID. Cette valeur doit être transmise dans le champ d'identification d'entité (faisant partie du champ **wIndex**) de chaque demande adressée à l'unité de fonction.

Le champ **bSourceID** est utilisé pour décrire la connectivité de cette unité SRC. Il contient l'ID de l'unité ou du terminal auquel cette unité SRC est connectée au moyen de sa broche d'entrée. Le descripteur de grappe, qui décrit les canaux logiques entrant dans l'unité SRC, n'est pas répété ici. Il appartient au logiciel hôte de tracer la connexion "en amont" pour localiser le descripteur de grappe relatif à cette grappe.

Le champ **bCSourceInID** contient une constante indiquant à quelle entité d'horloge la broche d'entrée d'horloge associée à la broche d'entrée audio est connectée.

Le champ **bCSourceOutID** contient une constante indiquant à quelle entité d'horloge la broche d'entrée d'horloge associée à la broche de sortie audio est connectée.

L'ID d'un descripteur de chaîne est fourni pour décrire plus précisément l'unité SRC.

Le tableau suivant présente les grandes lignes du descripteur d'unité SRC.

Tableau 4-32: Descripteur d'unité de conversion du débit d'échantillonnage

Décalage	Champ	Taille	Valeur	Description
0	bLength	1	Nombre	Taille de ce descripteur, en octets: 9.
1	bDescriptorType	1	Constante	Type de descripteur CS_INTERFACE.
2	bDescriptorSubtype	1	Constante	Sous-type de descripteur SAMPLE_RATE_CONVERTER.
3	bUnitID	1	Nombre	Valeur identifiant de manière unique l'unité au sein de la fonction audio. Cette valeur est utilisée dans toutes les demandes d'adressage de cette unité.
4	bSourceID	1	Nombre	ID de l'unité ou du terminal auquel cette unité SRC est connectée.
5	bCSourceInID	1	Nombre	ID de l'entité d'horloge à laquelle la section d'entrée de l'unité SRC est connectée.
6	bCSourceOutID	1	Nombre	ID de l'entité d'horloge à laquelle la section de sortie de l'unité SRC est connectée.
7	wSRCDescrStr	2	wStrDescrID	ID d'un descripteur de chaîne spécifique à la classe, décrivant l'unité SRC.

4.5.2.9 DESCRIPTEUR D'UNITÉ D'EFFET

L'unité d'effet est identifiée de manière unique par la valeur du champ **bUnitID** du descripteur d'unité d'effet (EUD). Aucun autre terminal ou unité au sein de l'interface AudioControl ne peut avoir le même ID. Cette valeur doit être transmise dans le champ d'identification d'entité (faisant partie du champ **wIndex**) de chaque demande adressée à l'unité d'effet.

Le champ **wEffectType** contient une valeur qui identifie entièrement l'unité d'effet. Pour obtenir la liste de tous les types d'unités d'effet pris en charge, voir A.19, "Types d'effets d'unité d'effet".

Le champ **bSourceID** est utilisé pour décrire la connectivité de cette unité d'effet. Il contient l'ID de l'unité ou du terminal auquel cette unité d'effet est connectée au moyen de sa broche d'entrée. Le descripteur de grappe, qui décrit les canaux logiques entrant dans l'unité d'effet, n'est pas répété ici. Il appartient au logiciel hôte de tracer la connexion "en amont" pour localiser le descripteur de grappe relatif à cette grappe.

bmControls() est une matrice à (ch+1) éléments de topogrammes binaires de 4 octets, suivant chacun la même sémantique que le champ **bmControls** dans d'autres descripteurs.

L'ID d'un descripteur de chaîne est fourni pour décrire plus précisément l'unité d'effet.

Le tableau suivant présente le descripteur d'unité d'effet.

Tableau 4-33: Descripteur d'unité d'effet

Décalage	Champ	Taille	Valeur	Description
0	bLength	1	Nombre	Taille de ce descripteur, en octets: 17+(ch*4).
1	bDescriptorType	1	Constante	Type de descripteur CS_INTERFACE.
2	bDescriptorSubtype	1	Constante	Sous-type de descripteur EFFECT_UNIT.
3	bUnitID	1	Nombre	Valeur identifiant de manière unique l'unité au sein de la fonction audio. Cette valeur est utilisée dans toutes les demandes d'adressage de cette unité.
4	wEffectType	2	Constante	Constante identifiant le type d'effet produit par cette unité.
6	bSourceID	1	Nombre	ID de l'unité ou du terminal auquel cette unité d'effet est connectée.
7	bmaControls(0)	4	Topogramme binaire	Topogramme binaire des commandes du canal maître 0: D31..0: Allocation spécifique à l'effet.
11	bmaControls(1)	4	Topogramme binaire	Topogramme binaire des commandes du canal 1: D31..0: Allocation spécifique à l'effet.
...	...	...	...	...
11+(ch*4)	bmaControls(ch)	4	Topogramme binaire	Topogramme binaire des commandes du canal ch: D31..0: Allocation spécifique à l'effet.
15+(ch*4)	wEffectsDescrStr	2	wStrDescrID	ID d'un descripteur de chaîne spécifique à la classe, décrivant cette unité d'effet.

4.5.2.9.1 DESCRIPTEUR D'UNITÉ D'EFFET DE SECTION D'EGALISEUR PARAMETRIQUE

Le champ **wEffectType** du descripteur d'unité d'effet commun contient la valeur PARAM_EQ_SECTION_EFFECT. (Voir A.19, "Types d'effets d'unité d'effet".)

Le tableau suivant présente le descripteur d'unité d'effet PEQS. Il est identique au descripteur d'unité d'effet commun, à l'exception de certaines valeurs de champ. Il est répété ici pour plus de clarté.

La commande d'activation doit toujours être présente et mise en œuvre en lecture-écriture. Elle n'est donc pas représentée dans le champ **bmControls**.

Tableau 4-34: Descripteur d'unité d'effet de section d'égaliseur paramétrique

Décalage	Champ	Taille	Valeur	Description
0	bLength	1	Nombre	Taille de ce descripteur, en octets: 17+(ch*4)
1	bDescriptorType	1	Constante	Type de descripteur CS_INTERFACE.
2	bDescriptorSubtype	1	Constante	Sous-type de descripteur EFFECT_UNIT.
3	bUnitID	1	Nombre	Valeur identifiant de manière unique l'unité au sein de la fonction audio. Cette valeur est utilisée dans toutes les demandes d'adressage de cette unité.
4	wEffectType	2	Constante	Type d'effet PARAM_EQ_SECTION_EFFECT.
6	bSourceID	1	Nombre	ID de l'unité ou du terminal auquel cette unité d'effet est connectée.
7	bmaControls(0)	4	Topogramme binaire	Topogramme binaire des commandes du canal maître 0: D1..0: Commande de fréquence médiane. D3..2: Commande de facteur Q. D5..4: Commande de gain. D7..6: Commande d'engorgement. D9..8: Commande de débordement. D31..10: Réservé.
11	bmaControls(1)	4	Topogramme binaire	Topogramme binaire du canal logique 1.
...	...	...	...	...
11+(ch*4)	bmaControls(ch)	4	Topogramme binaire	Topogramme binaire des commandes du canal logique ch.
15+(ch*4)	wEffectsDescrStr	2	wStrDescrID	ID d'un descripteur de chaîne spécifique à la classe, décrivant cette unité d'effet.

4.5.2.9.2 DESCRIPTEUR D'UNITE D'EFFET DE REVERBERATION

Le champ **wEffectType** du descripteur d'unité d'effet commun contient la valeur REVERBERATION_EFFECT. (Voir A.19, "Types d'effets d'unité d'effet".)

Le tableau suivant présente le descripteur d'unité d'effet de réverbération. Il est identique au descripteur d'unité d'effet commun, à l'exception de certaines valeurs de champ. Il est répété ici pour plus de clarté.

La commande d'activation doit toujours être présente et mise en œuvre en lecture-écriture. Elle n'est donc pas représentée dans le champ **bmControls**.

Tableau 4-35: Descripteur d'unité d'effet de réverbération

Décalage	Champ	Taille	Valeur	Description
0	bLength	1	Nombre	Taille de ce descripteur, en octets: 17+(ch*4).
1	bDescriptorType	1	Constante	Type de descripteur CS_INTERFACE.
2	bDescriptorSubtype	1	Constante	Sous-type de descripteur EFFECT_UNIT.
3	bUnitID	1	Nombre	Valeur identifiant de manière unique l'unité au sein de la fonction audio. Cette valeur est utilisée dans toutes les demandes d'adressage de cette unité.
4	wEffectType	2	Constante	Type d'effet REVERBERATION_EFFECT.
6	bSourceID	1	Nombre	ID de l'unité ou du terminal auquel cette unité d'effet est connectée.
7	bmaControls(0)	4	Topogramme binaire	Topogramme binaire des commandes du canal maître 0: D1..0: Commande de type. D3..2: Commande de niveau. D5..4: Commande de temps. D7..6: Commande de réaction de retard. D9..8: Commande de pré-retard. D11..10: Commande de densité. D13..12: Commande d'affaiblissement haute fréquence. D15..14: Commande d'engorgement. D17..16: Commande de débordement. D31..18: Réserve.
11	bmaControls(1)	4	Topogramme binaire	Topogramme binaire du canal logique 1.
...	...	...	...	...
11+(ch*4)	bmaControls(ch)	4	Topogramme binaire	Topogramme binaire des commandes du canal logique ch.
15+(ch*4)	wEffectsDescrStr	2	wStrDescrID	ID d'un descripteur de chaîne spécifique à la classe, décrivant cette unité d'effet.

4.5.2.9.3 DESCRIPTEUR D'UNITE D'EFFET DE RETARD DE MODULATION

Le champ **wEffectType** du descripteur d'unité d'effet commun contient la valeur MOD_DELAY_EFFECT.
(Voir A.19, "Types d'effets d'unité d'effet".)

Le tableau suivant présente le descripteur d'unité d'effet de retard de modulation. Il est identique au descripteur d'unité d'effet commun, à l'exception de certaines valeurs de champ. Il est répété ici pour plus de clarté.

Tableau 4-36: Descripteur d'unité d'effet de retard de modulation

Décalage	Champ	Taille	Valeur	Description
0	bLength	1	Nombre	Taille de ce descripteur, en octets: 17+(ch*4).
1	bDescriptorType	1	Constante	Type de descripteur CS_INTERFACE.
2	bDescriptorSubtype	1	Constante	Sous-type de descripteur EFFECT_UNIT.
3	bUnitID	1	Nombre	Valeur identifiant de manière unique l'unité au sein de la fonction audio. Cette valeur est utilisée dans toutes les demandes d'adressage de cette unité.
4	wEffectType	2	Constante	Type d'effet MOD_DELAY_EFFECT.
6	bSourceID	1	Nombre	ID de l'unité ou du terminal auquel cette unité d'effet est connectée.
7	bmaControls(0)	4	Topogramme binaire	Topogramme binaire des commandes du canal maître 0: D1..0: Commande d'activation. D3..2: Commande d'équilibrage. D5..4: Commande de débit. D7..6: Commande de profondeur. D9..8: Commande de temps. D11..10: Commande de niveau de rétroaction. D13..12: Commande d'engorgement. D15..14: Commande de débordement. D31..16: Réserve.
11	bmaControls(1)	4	Topogramme binaire	Topogramme binaire du canal logique 1.
...	...	...	...	...
11+(ch*4)	bmaControls(ch)	4	Topogramme binaire	Topogramme binaire des commandes du canal logique ch.
15+(ch*4)	wEffectsDescrStr	2	wStrDescrID	ID d'un descripteur de chaîne spécifique à la classe, décrivant cette unité d'effet.

4.5.2.9.4 DESCRIPTEUR D'UNITÉ D'EFFET DE COMPRESSEUR DE PLAGE DYNAMIQUE

Le champ **wEffectType** du descripteur d'unité d'effet commun contient la valeur DYN_RANGE_COMP_EFFECT. (Voir A.19, "Types d'effets d'unité d'effet".)

Le tableau suivant présente le descripteur d'unité d'effet de compresseur de plage dynamique. Il est identique au descripteur d'unité d'effet commun, à l'exception de certaines valeurs de champ. Il est répété ici pour plus de clarté.

Tableau 4-37: Descripteur d'unité d'effet de compresseur de plage dynamique

Décalage	Champ	Taille	Valeur	Description
0	bLength	1	Nombre	Taille de ce descripteur, en octets: 17+(ch*4).
1	bDescriptorType	1	Constante	Type de descripteur CS_INTERFACE.
2	bDescriptorSubtype	1	Constante	Sous-type de descripteur EFFECT_UNIT.
3	bUnitID	1	Nombre	Valeur identifiant de manière unique l'unité au sein de la fonction audio. Cette valeur est utilisée dans toutes les demandes d'adressage de cette unité.
4	wEffectType	2	Constante	Type d'effet DYN_RANGE_COMP_EFFECT.
6	bSourceID	1	Nombre	ID de l'unité ou du terminal auquel cette unité d'effet est connectée.
7	bmaControls(0)	4	Topogramme binaire	Topogramme binaire des commandes du canal maître 0. D1..0: Commande d'activation. D3..2: Commande de taux de compression. D5..4: Commande MaxAmpl. D7..6: Commande de seuil. D9..8: Commande de temps d'attaque. D11..10: Commande de temps de relâchement. D13..12: Commande d'engorgement. D15..14: Commande de débordement. D31..16: Réservé.
11	bmaControls(1)	4	Topogramme binaire	Topogramme binaire du canal logique 1.
...	...	...	...	...
11+(ch*4)	bmaControls(ch)	4	Topogramme binaire	Topogramme binaire des commandes du canal logique ch.
15+(ch*4)	wEffectsDescrStr	2	wStrDescrID	ID d'un descripteur de chaîne spécifique à la classe, décrivant cette unité d'effet.

4.5.2.10 DESCRIPTEUR D'UNITE DE TRAITEMENT

L'unité de traitement est identifiée de manière unique par la valeur du champ **bUnitID** du descripteur d'unité de traitement (PUD). Aucun autre terminal ou unité au sein de l'interface AudioControl ne peut avoir le même ID. Cette valeur doit être transmise dans le champ d'identification d'entité (faisant partie du champ **wIndex**) de chaque demande adressée à l'unité de traitement.

Le champ **wProcessType** contient une valeur qui identifie entièrement l'unité d'effet. Pour obtenir la liste de tous les types d'unités de traitement pris en charge, voir A.20, "Types de processus d'unité de traitement".

Le champ **bNrInPins** contient le nombre de broches d'entrée (p) de l'unité de traitement. La connectivité des broches d'entrée est décrite au moyen de la matrice **baSourceID()**, qui contient p éléments. L'indice i de la matrice est à base un et est directement lié aux numéros des broches d'entrée. **baSourceID(i)** contient l'ID de l'unité ou du terminal auquel la broche d'entrée i est connectée. Les descripteurs de grappe, qui décrivent les canaux logiques entrant dans l'unité de traitement, ne sont pas répétés ici. Il appartient au logiciel hôte de tracer les connexions "en amont" afin de localiser les descripteurs de grappe relatifs aux grappes.

Comme une unité de traitement peut librement redéfinir les localisations spatiales des canaux de sortie logiques contenus dans sa grappe de sortie, un descripteur de grappe de sortie est nécessaire. Le champ **wClusterDescrID** contient l'ID unique du descripteur de grappe qui caractérise la grappe quittant l'unité de traitement par la broche de sortie unique (connexion "en aval"). Pour une description détaillée du descripteur de grappe, voir 4.3, "Descripteur de grappe".

Une unité de traitement ne doit pas prendre en charge une commande d'activation. Si la fonctionnalité de la fonction audio exige que l'unité de traitement puisse être contournée dans certains scénarios, la topologie de la fonction doit l'expliquer par l'utilisation d'une unité de sélection, une broche d'entrée de l'unité de sélection étant connectée à la broche de sortie de l'unité de traitement et l'autre broche d'entrée de l'unité de sélection étant connectée à une entité qui fournit une grappe de canaux de sortie compatible avec la grappe de sortie de l'unité de traitement. Il peut s'agir de la même broche de sortie à laquelle la broche d'entrée de l'unité de traitement est connectée (contournement direct) si la configuration de grappe n'est pas modifiée par l'unité de traitement.

En général, toutes les commandes sont facultatives. Toutefois, certains types de traitements peuvent définir des commandes obligatoires.

L'ID d'un descripteur de chaîne est fourni pour décrire plus précisément l'unité de traitement.

Les champs précédents sont communs à toutes les unités de traitement. Cependant, en fonction de la valeur du champ **wProcessType**, une partie spécifique au processus est ajoutée au descripteur. Les alinéas qui suivent décrivent ces parties spécifiques au processus.

Le tableau suivant présente la partie commune du descripteur d'unité de traitement.

Tableau 4-38: Partie commune du descripteur d'unité de traitement

Décalage	Champ	Taille	Valeur	Description
0	bLength	1	Nombre	Taille de ce descripteur, en octets: 9+p+x.
1	bDescriptorType	1	Constante	Type de descripteur CS_INTERFACE.
2	bDescriptorSubtype	1	Constante	Sous-type de descripteur PROCESSING_UNIT.
3	bUnitID	1	Nombre	Valeur identifiant de manière unique l'unité au sein de la fonction audio. Cette valeur est utilisée dans toutes les demandes d'adressage de cette unité.
4	wProcessType	2	Constante	Constante identifiant le type de processus exécuté par cette unité.
6	bNrInPins	1	Nombre	Nombre de broches d'entrée de cette unité: p
7	baSourceID(1)	1	Nombre	ID de l'unité ou du terminal auquel la première broche d'entrée de cette unité de traitement est connectée.
...	...	...	...	...
7+(p-1)	baSourceID (p)	1	Nombre	ID de l'unité ou du terminal auquel la dernière broche d'entrée de cette unité de traitement est connectée.
7+p	wProcessingDescrStr	2	wStrDescrID	ID d'un descripteur de chaîne spécifique à la classe, décrivant cette unité de traitement.
9+p	Spécifique au processus	x	N/A	Un descripteur spécifique au processus est ajouté au descripteur commun. Voir les alinéas suivants.

4.5.2.10.1 DESCRIPTEUR D'UNITÉ DE TRAITEMENT DE MIXAGE ÉLEVATEUR/REDUCTEUR

Le champ **wProcessType** du descripteur d'unité de traitement commun contient la valeur UP/DOWNMIX_PROCESS. (Voir A.20, "Types de processus d'unité de traitement".)

L'unité de traitement de mixage élévateur/réducteur dispose d'une broche d'entrée unique. Par conséquent, le champ **bNrInputs** doit contenir la valeur 1.

La commande de choix de mode (D1..0) est utilisée pour modifier le comportement de l'unité de traitement en choisissant différents modes de fonctionnement.

Le descripteur spécifique au processus de l'unité de traitement de mixage élévateur/réducteur décrit les modes de fonctionnement pris en charge par l'unité de traitement. Le choix d'un mode de fonctionnement s'effectue en émettant la demande "Set Mode Select Control". Le nombre de modes pris en charge (m) est contenu dans le champ **bNrModes**. Ce champ est suivi d'une matrice de champs d'ID de descripteur de grappe, **waClusterDescrID(i)**. L'indice i de cette matrice est à base un et est directement lié au numéro du mode décrit par l'entrée **waClusterDescrID(i)**. La valeur i doit être utilisée comme paramètre pour la demande "Set Mode" afin de choisir le mode i .

Chaque champ **waClusterDescrID(i)** contient l'ID unique d'un descripteur de grappe décrivant la configuration de grappe de sortie lorsque le mode i est sélectionné. La modification du mode par l'hôte au moyen d'une demande "Set Mode Select Control" entraîne un changement de descripteur de grappe de sortie. Toutefois, cela ne doit pas entraîner d'interruption du descripteur haute capacité, l'hôte lui-même ayant initié la modification.

Le tableau suivant présente la combinaison des descripteurs d'unité de traitement de mixage élévateur/réducteur communs et spécifiques au processus.

Tableau 4-39: Descripteur d'unité de traitement de mixage élévateur/réducteur

Décalage	Champ	Taille	Valeur	Description
0	bLength	1	Nombre	Taille de ce descripteur, en octets: 15+2*m.
1	bDescriptorType	1	Constante	Type de descripteur CS_INTERFACE.
2	bDescriptorSubtype	1	Constante	Sous-type de descripteur PROCESSING_UNIT.
3	bUnitID	1	Nombre	Valeur identifiant de manière unique l'unité au sein de la fonction audio. Cette valeur est utilisée dans toutes les demandes d'adressage de cette unité.
4	wProcessType	2	Constante	Type de processus UP/DOWNMIX_PROCESS.
6	bNrInPins	1	Nombre	Nombre de broches d'entrée de cette unité: 1
7	baSourceID(1)	1	Nombre	ID de l'unité ou du terminal auquel la broche d'entrée de cette unité de traitement est connectée.
8	wProcessingDescrStr	2	wStrDescrID	ID d'un descripteur de chaîne spécifique à la classe, décrivant cette unité de traitement.
10	bmControls	4	Topogramme binaire	D1..0: Commande de choix de mode. D3..2: Commande d'engorgement. D5..4: Commande de débordement. D31..6: Réservé.
14	bNrModes	1	Nombre	Nombre de modes pris en charge par cette unité de traitement: m
15	waClusterDescrID(1)	2	Nombre	ID unique du descripteur de grappe pour le mode(1).
...	...	...	...	...
15+2*(m-1)	waClusterDescrID(m)	2	Nombre	ID unique du descripteur de grappe pour le mode(m).

4.5.2.10.2 DESCRIPTEUR D'UNITE DE TRAITEMENT D'AMPLIFICATION STEREO

Le champ **wProcessType** du descripteur d'unité de traitement commun contient la valeur STEREO_EXTENDER_PROCESS. (Voir A.20, "Types de processus d'unité de traitement".)

L'unité de traitement d'amplification stéréo dispose d'une broche d'entrée unique. Par conséquent, le champ **bNrInputs** doit contenir la valeur 1.

La grappe d'entrée de l'unité de traitement d'amplification stéréo doit contenir uniquement les canaux d'entrée logiques gauche et droite. La grappe de sortie doit donc également ne contenir que les canaux logiques gauche et droite. Par conséquent, un descripteur de grappe de sortie n'est pas nécessaire pour cette unité de traitement.

Il n'existe pas de descripteur spécifique au processus pour l'unité de traitement d'amplification stéréo.

Le tableau suivant présente le descripteur d'unité de traitement d'amplification stéréo.

Tableau 4-40: Descripteur d'unité de traitement d'amplification stéréo

Décalage	Champ	Taille	Valeur	Description
0	bLength	1	Nombre	Taille de ce descripteur, en octets: 14.
1	bDescriptorType	1	Constante	Type de descripteur CS_INTERFACE.
2	bDescriptorSubtype	1	Constante	Sous-type de descripteur PROCESSING_UNIT.
3	bUnitID	1	Nombre	Valeur identifiant de manière unique l'unité au sein de la fonction audio. Cette valeur est utilisée dans toutes les demandes d'adressage de cette unité.
4	wProcessType	2	Constante	Type de processus STEREO_EXTENDER_PROCESS.
6	bNrInPins	1	Nombre	Nombre de broches d'entrée de cette unité: 1
7	baSourceID(1)	1	Nombre	ID de l'unité ou du terminal auquel la broche d'entrée de cette unité de traitement est connectée.
8	wProcessingDescrStr	2	wStrDescrID	ID d'un descripteur de chaîne spécifique à la classe, décrivant cette unité de traitement.
10	bmControls	4	Topogramme binaire	D1..0: Commande de largeur. D3..2: Commande d'engorgement. D5..4: Commande de débordement. D31..6: Réservé.

4.5.2.10.3 DESCRIPTEUR D'UNITE DE TRAITEMENT MULTIFONCTION

Le champ **wProcessType** du descripteur d'unité de traitement contient la valeur MULTI_FUNCTION_PROCESS. (Voir A.20, "Types de processus d'unité de traitement".)

L'unité de traitement multifonction peut avoir plusieurs broches d'entrée, comme indiqué dans le champ **bNrInPins**.

Comme une unité de traitement multifonction peut librement redéfinir sa configuration de grappe de sortie, un descripteur de grappe de sortie est nécessaire. Le champ **wClusterDescrID** contient l'ID unique du descripteur de grappe qui caractérise la grappe quittant l'unité de traitement par sa broche de sortie unique (connexion "en aval"). Pour une description détaillée du descripteur de grappe, voir 4.3, "Descripteur de grappe".

Le champ **bmAlgorithms** est un topogramme binaire indiquant les types d'algorithmes exécutés au sein de l'unité de traitement multifonction. Plusieurs bits peuvent être définis simultanément.

Le tableau suivant présente le descripteur d'unité de traitement multifonction.

Tableau 4-41: Descripteur d'unité de traitement multifonction

Décalage	Champ	Taille	Valeur	Description
0	bLength	1	Nombre	Taille de ce descripteur, en octets: 19+p.
1	bDescriptorType	1	Constante	Type de descripteur CS_INTERFACE.
2	bDescriptorSubtype	1	Constante	Sous-type de descripteur PROCESSING_UNIT.
3	bUnitID	1	Nombre	Valeur identifiant de manière unique l'unité au sein de la fonction audio. Cette valeur est utilisée dans toutes les demandes d'adressage de cette unité.
4	wProcessType	2	Constante	Type de processus MULTI_FUNCTION_PROCESS.
6	bNrInPins	1	Nombre	Nombre de broches d'entrée de cette unité: 1
7	baSourceID(1)	1	Nombre	ID de l'unité ou du terminal auquel la broche d'entrée 1 de cette unité de traitement est connectée.
...	...	...	...	...
7+(p-1)	baSourceID (p)	1	Nombre	ID de l'unité ou du terminal auquel la dernière broche d'entrée de cette unité de traitement est connectée.
7+p	wProcessingDescrStr	2	wStrDescrID	ID d'un descripteur de chaîne spécifique à la classe, décrivant cette unité de traitement.
9+p	bmControls	4	Topogramme binaire	D1..0: Commande d'engorgement. D3..2: Commande de débordement. D31..4: Réservé.
13+p	wClusterDescrID	2	Nombre	ID unique du descripteur de grappe pour cette unité de traitement.
15+p	bmAlgorithms	4	Topogramme binaire	D0: Algorithme non défini. D1: Formation de faisceau. D2: Annulation d'écho acoustique. D3: Annulation active du bruit. D4: Séparation de source aveugle. D5: Suppression/réduction du bruit. D31..6: Réservé.

4.5.2.11 DESCRIPTEUR D'UNITÉ D'EXTENSION

L'unité d'extension est identifiée de manière unique par la valeur du champ **bUnitID** du descripteur d'unité d'extension (XUD). Aucun autre terminal ou unité au sein de l'interface AudioControl ne peut avoir le même ID. Cette valeur doit être transmise dans le champ d'identification d'entité (faisant partie du champ **wIndex**) de chaque demande adressée à l'unité d'extension.

Le descripteur d'unité d'extension fournit des informations minimales sur l'unité d'extension pour un pilote générique, au moins pour signaler la présence de composants spécifiques au fournisseur au sein de la fonction audio. Le champ **wExtensionCode** peut contenir un code spécifique au fournisseur qui identifie plus précisément l'unité d'extension. S'il n'est pas utilisé, il convient de le définir sur zéro.

Le champ **bNrInPins** contient le nombre de broches d'entrée (p) de l'unité d'extension. La connectivité des broches d'entrée est décrite au moyen de la matrice **baSourceID()**, qui contient p éléments. L'indice i de la matrice est à base un et est directement lié aux numéros des broches d'entrée. **baSourceID(i)** contient l'ID de l'unité ou du terminal auquel la broche d'entrée i est connectée. Les descripteurs de grappe, qui décrivent les canaux logiques entrant dans l'unité d'extension, ne sont pas répétés ici. Il appartient au logiciel hôte de tracer les connexions "en amont" afin de localiser les descripteurs de grappe relatifs aux grappes.

Comme une unité d'extension peut librement redéfinir sa configuration de grappe de sortie, un descripteur de grappe de sortie est nécessaire. Le champ **wClusterDescrID** contient l'ID unique du descripteur de grappe qui caractérise la grappe quittant l'unité d'extension par sa broche de sortie unique (connexion "en aval"). Pour une description détaillée du descripteur de grappe, voir 4.3, "Descripteur de grappe".

L'ID d'un descripteur de chaîne est fourni pour décrire plus précisément l'unité d'extension.

Le tableau suivant décrit le descripteur d'unité d'extension.

Tableau 4-42: Descripteur d'unité d'extension

Décalage	Champ	Taille	Valeur	Description
0	bLength	1	Nombre	Taille de ce descripteur, en octets: 15+p.
1	bDescriptorType	1	Constante	Type de descripteur CS_INTERFACE.
2	bDescriptorSubtype	1	Constante	Sous-type de descripteur EXTENSION_UNIT.
3	bUnitID	1	Nombre	Valeur identifiant de manière unique l'unité au sein de la fonction audio. Cette valeur est utilisée dans toutes les demandes d'adressage de cette unité.
4	wExtensionCode	2	Constante	Code spécifique au fournisseur identifiant l'unité d'extension.
6	bNrInPins	1	Nombre	Nombre de broches d'entrée de cette unité: p
7	baSourceID(1)	1	Nombre	ID de l'unité ou du terminal auquel la première broche d'entrée de cette unité d'extension est connectée.
...	...	...	...	...
7+(p-1)	baSourceID(p)	1	Nombre	ID de l'unité ou du terminal auquel la dernière broche d'entrée de cette unité d'extension est connectée.
7+p	wExtensionDescrStr	2	wStrDescrID	ID d'un descripteur de chaîne spécifique à la classe, décrivant cette unité d'extension.
9+p	bmControls	4	Topogramme binaire	D1..0: Commande d'engorgement. D3..2: Commande de débordement. D31..4: Réservé.
13+p	wClusterDescrID	2	Nombre	ID unique du descripteur de grappe pour cette unité d'extension.

4.5.2.12 DESCRIPTEUR DE SOURCE D'HORLOGE

L'entité source d'horloge est identifiée de manière unique par la valeur du champ **bClockID** du descripteur d'entité source d'horloge (CSD). Aucun autre terminal, entité d'horloge ou unité au sein de l'interface AudioControl ne peut avoir le même ID. Cette valeur doit être transmise dans le champ d'identification d'entité (faisant partie du champ **wIndex**) de chaque demande adressée à l'entité source d'horloge.

Le champ **bmAttributes** contient un champ de bits de type d'horloge (D0) qui indique si la source d'horloge représente une horloge externe (D0 = 0b0) ou une horloge interne (D0=0b1). D'autres caractéristiques de la source d'horloge peuvent être dérivées de la valeur de la paire de bits de commande de fréquence d'horloge D1..0 dans le champ **bmControls**. Comme la commande de fréquence d'horloge doit toujours être présente, les seules valeurs admises sont D1..0 = 0b01 (lecture seule) ou D1..0 = 0b11 (lecture-écriture). Les fréquences d'horloge prises en charge peuvent être dérivées de l'attribut de plage de la commande de fréquence d'horloge (débit fixe ou variable). Noter que même une source d'horloge de type externe peut être en lecture-écriture si la fonction audio a la capacité d'influencer cette horloge externe par des moyens hors USB. La fréquence d'échantillonnage réelle de la source d'horloge peut être manipulée à l'aide de la demande de fréquence d'horloge. De plus, la source d'horloge peut être interrogée quant à la validité de son signal d'horloge d'échantillonnage actuel par le biais d'une demande "Get Clock Validity".

Le bit D1 du champ **bmAttributes** indique si une horloge interne est en fréquence indépendante (D1 = 0b0) ou synchronisée au début de trame (D1 = 0b1). Si D0 = 0b0, alors D1 doit également être défini sur 0b0.

Le champ **bReferenceTerminal** contient une référence à un terminal à partir duquel la source d'horloge est dérivée. Cela est par exemple utile lorsque le signal d'horloge d'une source d'horloge est dérivé du signal d'entrée sur un connecteur S/PDIF, représenté par un terminal d'entrée. Si la source d'horloge est en fréquence indépendante ou dérivée du SOF USB (non dérivée d'un terminal), ce champ doit être défini sur zéro.

L'ID d'un descripteur de chaîne est fourni pour décrire plus précisément l'unité source d'horloge.

Le tableau suivant présente les grandes lignes du descripteur de source d'horloge.

Tableau 4-43: Descripteur de source d'horloge

Décalage	Champ	Taille	Valeur	Description
0	bLength	1	Nombre	Taille de ce descripteur, en octets: 12.
1	bDescriptorType	1	Constante	Type de descripteur CS_INTERFACE.
2	bDescriptorSubtype	1	Constante	Sous-type de descripteur CLOCK_SOURCE.
3	bClockID	1	Nombre	Valeur identifiant de manière unique l'entité source d'horloge au sein de la fonction audio. Cette valeur est utilisée dans toutes les demandes d'adressage de cette entité.
4	bmAttributes	1	Topogramme binaire	D0: Type d'horloge: 0: Horloge externe. 1: Horloge interne. D1: Type de synchronisation: 0: Asynchrone. 1: Horloge synchronisée au SOF. D7..2: Réserve.
5	bmControls	4	Topogramme binaire	D1..0: Commande de fréquence d'horloge. D3..2: Commande de validité d'horloge. D31..4: Réserve.
9	bReferenceTerminal	1	Nombre	ID de terminal du terminal à partir duquel cette source d'horloge est dérivée.
10	wClockSourceStr	2	wStrDescrID	ID d'un descripteur de chaîne spécifique à la classe, décrivant l'entité source d'horloge.

4.5.2.13 DESCRIPTEUR DE SELECTEUR D'HORLOGE

L'entité de sélecteur d'horloge est identifiée de manière unique par la valeur du champ **bClockID** du descripteur d'entité de sélecteur d'horloge (CXD). Aucun autre terminal, entité d'horloge ou unité au sein de l'interface AudioControl ne peut avoir le même ID. Cette valeur doit être transmise dans le champ d'identification d'entité (faisant partie du champ **wIndex**) de chaque demande adressée à l'entité de sélecteur d'horloge.

Le champ **bNrInPins** contient le nombre de broches d'entrée d'horloge (p) de l'entité de sélecteur d'horloge. La connectivité des broches d'entrée est décrite au moyen de la matrice **baSourceID()**, qui contient p éléments. L'indice i de la matrice est à base un et est directement lié aux numéros des broches d'entrée d'horloge.

baSourceID(i) contient l'ID de l'entité d'horloge à laquelle la broche d'entrée d'horloge i est connectée.

L'ID d'un descripteur de chaîne est fourni pour décrire plus précisément l'entité de sélecteur d'horloge.

Le tableau suivant présente les grandes lignes du descripteur de sélecteur d'horloge.

Tableau 4-44: Descripteur de sélecteur d'horloge

Décalage	Champ	Taille	Valeur	Description
0	bLength	1	Nombre	Taille de ce descripteur, en octets: 11+p.
1	bDescriptorType	1	Constante	Type de descripteur CS_INTERFACE.
2	bDescriptorSubtype	1	Constante	Sous-type de descripteur CLOCK_SELECTOR.
3	bClockID	1	Nombre	Valeur identifiant de manière unique l'entité de sélecteur d'horloge au sein de la fonction audio. Cette valeur est utilisée dans toutes les demandes d'adressage de cette entité.
4	bNrInPins	1	Nombre	Nombre de broches d'entrée de cette unité: p
5	baSourceID(1)	1	Nombre	ID de l'entité d'horloge à laquelle la première broche d'entrée d'horloge de cette entité de sélecteur d'horloge est connectée.
...	...	...	...	...
5+(p-1)	baSourceID(p)	1	Nombre	ID de l'entité d'horloge à laquelle la dernière broche d'entrée d'horloge de cette entité de sélecteur d'horloge est connectée.
5+p	bmControls	4	Topogramme binaire	D1..0: Commande de sélecteur d'horloge. D31..2: Réserve.
9+p	wSelectorDescrStr	2	wStrDescrID	ID d'un descripteur de chaîne spécifique à la classe, décrivant l'entité de sélecteur d'horloge.

4.5.2.14 DESCRIPTEUR DE MULTIPLICATEUR D'HORLOGE

L'entité de multiplicateur d'horloge est identifiée de manière unique par la valeur du champ **bClockID** du descripteur d'entité de multiplicateur d'horloge (CMD). Aucun autre terminal, entité d'horloge ou unité au sein de l'interface AudioControl ne peut avoir le même ID. Cette valeur doit être transmise dans le champ d'identification d'entité (faisant partie du champ **wIndex**) de chaque demande adressée à l'entité de multiplicateur d'horloge.

Le champ **bSourceID** contient l'ID de l'entité d'horloge à laquelle la broche d'entrée d'horloge du multiplicateur d'horloge est connectée.

L'ID d'un descripteur de chaîne est fourni pour décrire plus précisément l'entité de multiplicateur d'horloge.