Universal Serial Bus
Device Class Definition
for
Audio/Video Devices

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

AV Device Class Overview
&
AVFunction Definition

Release 1.0

December 07, 2011

Scope of This Release

This document is the Release 1.0 of this Device Class Definition.

Contributors

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

Will Harris	Texas Instruments
Grant Ley	Texas Instruments

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

Table of Contents

List of Tables

List of Figures

1. Introduction

1.1. Scope

The USB Audio/Video (AV) Device Class Definition describes the methods used to communicate with devices or functions embedded in composite devices that are used to manipulate audio, video, voice, and all image- and sound-related functionality. This includes both AV data (analog and digital) and associated metadata and the functionality that is used to directly control the AV environment, such as Volume and Contrast Controls. Examples range from cameras used for video chat and conferencing to full-fledged TV sets that can be used as displays for mobile devices.

The AV Device Class does not include functionality to operate transport mechanisms that are related to the reproduction of AV data, such as tape transport mechanisms or CD-ROM drive control. Handling of MIDI data streams over the USB is directly related to audio but will not be covered in this version of the specification. Future versions may include support for MIDI.

It should also be noted that, although one of the main focuses of this specification is to enable USB-connected Video Display Devices, the USB is not intended to be a substitution for dedicated video connections, such as HDMI® or DisplayPort™. By nature, USB is a shared bus with shared bandwidth and resources among all participants on the bus whereas HDMI and DisplayPort are dedicated point-to-point connections with predefined bandwidth availability. Also, the USB uses a control plane model and data streaming models that are substantially different from those found on dedicated video bus architectures so that a USB-connected Video Display Device will present itself differently to a Controller than other Video Display Devices that reside on dedicated video buses.

1.2. Purpose

The purpose of this document is to describe the minimum capabilities and characteristics an AV function shall support to be compliant. This document also provides recommendations for optional features.

1.3. Related Documents

- [USB2.0] – Universal Serial Bus Specification, Revision 2.0, April 27, 2000 (referred to in this document as the USB 2.0 Specification) (available at: http://www.usb.org/developers/docs).
- [USB3.0] – Universal Serial Bus 3.0 Specification, Revision 1.0 (including errata and ECN's through May 1, 2011), June 6, 2011 (referred to in this document as the USB 3.0 Specification) (available at: http://www.usb.org/developers/docs.)
- [AUDIO1.0] – Universal Serial Bus Device Class Definition for Audio Devices, Release 1.0, March 18, 1998 (available at: http://www.usb.org/developers/devclass_docs).
- [FORMATS1.0] – Universal Serial Bus Device Class Definition for Audio Data Formats, Release 1.0, March 18, 1998 (available at: http://www.usb.org/developers/devclass_docs).
- [TERMTYPES1.0] – Universal Serial Bus Device Class Definition for Terminal Types, Release 1.0, March 18, 1998 (available at: http://www.usb.org/developers/devclass_docs).
- [AUDIO2.0] – Universal Serial Bus Device Class Definition for Audio Devices, Release 2.0, May 31, 2006 (available at: http://www.usb.org/developers/devclass_docs).
- [FORMATS2.0] – Universal Serial Bus Device Class Definition for Audio Data Formats, Release 2.0, May 31, 2006 (available at: http://www.usb.org/developers/devclass_docs).
- [TERMTYPES2.0] – Universal Serial Bus Device Class Definition for Terminal Types, Release 2.0, May 31, 2006 (available at: http://www.usb.org/developers/devclass_docs).
- [USBCS] – Universal Serial Bus Device Class Definition for Content Security Devices – Content Security Framework, Revision 2.0 (available at: http://www.usb.org/developers/devclass_docs).
- [USBCSM-5] – Universal Serial Bus Device Class Definition for Content Security Devices – Content Security Method 5 – High-bandwidth Digital Content Protection 2.1 (HDCP 2.1) Implementation, Revision 1.0 (available at: http://www.usb.org/developers/devclass_docs).
- USBECNIAD – USB Engineering Change Notice: Interface Association Descriptors (available at: http://www.usb.org/developers/docs).

- [USBIADDCC] – USB Interface Association Descriptor Device Class Code and Use Model, Revision 1.0, July 23, 2003 (available at: http://www.usb.org/developers/whitepapers).
- [USBLANGIDS] – Universal Serial Bus Language Identifiers (LANGIDs), Revision 1.0, March 29, 2000 (available at: http://www.usb.org/developers/docs).
- [AVFUNCTION] – Universal Serial Bus Device Class Definition for Audio/Video Devices – AV Device Class Overview & AVFunction Definition, Release 1.0, December 07, 2011 (available at: http://www.usb.org/developers/devclass_docs).
- [AVFORMAT_1] – Universal Serial Bus Device Class Definition for Audio/Video Devices – AVFormat 1 – Video over Bulk, Release 1.0, December 07, 2011 (available at: http://www.usb.org/developers/devclass_docs).
- [AVFORMAT_2] – Universal Serial Bus Device Class Definition for Audio/Video Devices – AVFormat 2 – Isochronous Audio, Release 1.0, December 07, 2011 (available at: http://www.usb.org/developers/devclass_docs).
- [AVFORMAT_3] – Universal Serial Bus Device Class Definition for Audio/Video Devices – AVFormat 3 – Uncompressed Full Frame Isochronous Video, Release 1.0, December 07, 2011 (available at: http://www.usb.org/developers/devclass_docs).
- [AVSCHEMA] – Available at: http://avschemas.usb.org/v1/avschema.xsd
- [BDP] – Universal Serial Bus Device Class Definition for Audio/Video Devices – Basic Device Profile, Release 1.0, (available at: http://www.usb.org/developers/whitepapers).
- [ANSIS1_11] – ANSI S1.11-2004 (R2009) standard (available at: http://www.ansi.org).
- [IEC11172_3] – MPEG-1 standard ISO/IEC 11172-3:1993 Information technology – Coding of moving pictures and associated audio for digital storage media at up to about 1.5 Mbit/s – Part 3: Audio (available at http://www.iec.ch).
- [IEC13818_1] – MPEG-2 standard ISO/IEC 13818:2000 Information technology – Generic coding of moving pictures and associated audio information – Part 1: Systems (available at http://www.iec.ch).
- [IEC13818_3] – MPEG-2 standard ISO/IEC 13818:1998 Information technology – Generic coding of moving pictures and associated audio information – Part 3: Audio" (available at http://www.iec.ch).
- [AC_3] – Digital Audio Compression Standard (AC-3, Enhanced AC-3), ETSI TS 102 366 (available at http://www.etsi.org).
- [IEEE_754] – ANSI/IEEE-754 floating-point standard (available at http://www.ieee.org).
- [IEC60958] – ISO/IEC 60958 International Standard: Digital Audio Interface and Annexes (available at: http://www.iec.ch).
- [IEC61937] – ISO/IEC 61937 standard (available at: http://www.iec.ch).
- [ETSI_TS_102_114] – ETSI Specification TS 102 114, "DTS Coherent Acoustics; Core and Extensions" (available at http://www.etsi.org).
- [HDCP2.1] – High-bandwidth Digital Content Protection System. Interface Independent Adaptation. Revision 2.1, July 18, 2011 (available from http://www.digital-cp.com).
- [MLP] – DVD Specifications for High Definition Video: MLP Reference Information.
- [IEC14496_3] – MPEG-4 Standard ISO/IEC 14496-3 – Information Technology – Coding of audio-visual objects – Part 3: Audio (available at http://www.iec.ch).
- [IEC14496_10] – MPEG-4 Standard ITU-T H.264 and ISO/IEC 14496-10:2004 – Information Technology – Coding of audio-visual objects – Part 10: Advanced Video Coding. Second Edition 2004-10-01 (available at http://www.iec.ch).
- [WMA] – Audio compression format from Microsoft. For technical and licensing information, contact Microsoft directly (http://www.microsoft.com/windows/windowsmedia/default.aspx).
- [HDMI] – The official High Definition Multimedia Interface website http://www.hdmi.org.
- [RFC5646] – Tags for Identifying Languages, September 2009 (available at http://www.rfc-editor.org/rfc/rfc5646.txt).
- [KHRONOS] – The Khronos Group, Open Standards for Media Authoring and Acceleration (available at http://www.khronos.org).
- [CEA-861-E] – A DTV Profile for Uncompressed High Speed Digital Interfaces, March 2008 (available at http://www.cea.org).
- [VESA] – Video Electronics Standards Association (available at http://www.vesa.org).

- [IEC10918_4] – JPEG Standard ISO/IEC 10918-4 – Information technology – Digital compression and coding of continuous-tone still images: Registration of JPEG profiles, SPIFF profiles, SPIFF tags, SPIFF colour spaces, APPn markers, SPIFF compression types and Registration Authorities (REGAUT). First Edition 1999-08-15 (available at http://www.iec.ch).

1.4. Terms and Abbreviations

This section defines terms and abbreviations used throughout this document. For additional terms and abbreviations that pertain to the Universal Serial Bus, see Chapter 2, "Terms and Abbreviations," in [USB2.0] and [USB3.0].

Table 1-1: Terms and Abbreviations

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

Term	Description
AVBundle	A group of physical VideoChannels AudioChannels and MetadataChannels that carry tightly related (from a content perspective) synchronous video, audio, and metadata information over a USB pipe.
AVCluster	A group of logical VideoChannels AudioChannels and MetadataChannels that carry tightly related (from a content perspective) synchronous video, audio, and metadata information over an interconnect within the AVCore.
AVCore	That part of the AVFunction that contains most of the building blocks (Terminals, Units) that makes up most of the AVControl functionality of the AVFunction. Explicitly excludes the AVData Streaming Interfaces that are part of the AVFunction.
AVControl Interface	A USB interface used to access the AVControls inside an AVFunction.
AVData Entity	An addressable Entity representing a source of AV data flowing into or a sink for AV data flowing out of the AVCore (including AVData Streaming Interfaces).
AVData Streaming Interface	A USB interface used to transport AVStreams into or out of the AVFunction.
AVFunction	An independent part of a USB device that deals with AV-related functionality. Includes the AVCore and all AVData Entities.
AVHeader	The Header present in the AudioPayload or VideoPayload of a Command or Notify Message. Also, the Header present in every SIP.
AVStream	A generic name for either a VideoStream or an AudioStream. See VideoStream and AudioStream.
AVStream Configuration	A generic name for either a VideoStream Configuration or an AudioStream Configuration. See VideoStream Configuration and AudioStream Configuration.
BDP	Basic Device Profile. See [BDP]
BIB	Bus Interval Boundary ([USB3.0]).
CBP	Control Bulk Pair. A pair of one Bulk IN and one Bulk OUT endpoint in the AVControl Interface that is used for AV class-specific messaging.
CBP Idle Condition	A condition when an AVFunction determines that it will not have data available on its CBP Bulk IN pipe for a certain amount of time (AV_CBP_IDLE_TIME) and informs the Host of this condition to avoid having the Host continually issue IN tokens for that pipe.
Channel	A logical channel in an AVCluster. Can be a VideoChannel, AudioChannel, or MetadataChannel.
Controllee	The entity that is controlled by the Controller. The AVFunction always assumes this role.
Controller	The entity that controls the AVFunction. In this version of the specification, the USB Host assumes this role.
Control Sequence	A sequence of a Command and Response Message that is used to convey or retrieve one or more control parameters to or from the AVFunction.
Converter Unit (CU)	Provides the means to transform an incoming VideoTrack and/or AudioTrack into another VideoTrack and/or AudioTrack with different characteristics.
CUD	Converter Unit Description.
Description	Section of the AVDD (AV Description Document) that provides a detailed XML description of a specific Entity or other building block of the AVFunction architecture. Replaces the concept of the class-specific USB Descriptor.
DTS	Digital Theater Systems.
DUD	Metadata Unit Description.
DVD	Digital Versatile Disc.
Effect Unit (EU)	Provides advanced AV manipulation on the incoming logical AudioChannels.
Encoded Audio Bit Stream	A concatenation of a potentially very large number of encoded audio frames, ordered according to ascending time.
Encoded AudioFrame	A sequence of bits that contains an encoded representation of AudioSamples from one or more physical audio channels taken over a fixed period of time.
Encoded VideoFrame	A sequence of bits that contains an encoded representation of VideoSamples from one VideoFrame.
Entity	An addressable logical object inside an AVFunction.
EUD	Effect Unit Description.
Feature Unit (FU)	Provides basic AV manipulation on the incoming logical AV channels.

Term	Description
FUD	Feature Unit Description.
GraphicsEngine Entity (GEE)	An addressable Entity inside the AVCore representing a graphics subsystem of the AVFunction.
H.264	ITU name of ISO MPEG4 part10 Advanced Video Coding standard. See [IEC14496_10].
HE_AAC	High Efficiency Advanced Audio Coding
Header	A collection of SubHeaders present in a SIP, containing Extended Audio Format data.
Input Pin	A logical input connection to an Entity. Carries a single AVCluster.
Input Terminal (IT)	A receptacle for AV information flowing into the AVFunction.
Inter-VideoFrame	A VideoFrame composed of Intra-coded and Inter-coded MacroBlocks.
Intra-VideoFrame	A VideoFrame composed of Intra-coded MacroBlocks only.
ITD	Input Terminal Description.
MacroBlock	An elementary group of VideoSamples considered by an H.264 encoder.
MetadataBundle	The group of all the metadata-only physical channels in an AVBundle.
(Logical) MetadataChannel	A logical transport medium for a single metadata channel. Makes abstraction of the physical Properties and formats of the connection. Is identified by MetadataChannel Type.
MetadataChannel Type	The metadata type of a MetadataChannel. Uniquely identifies the MetadataChannel. Examples are Subtitle (SUB), Commentary (COM), etc.
MetadataCluster	The group of all the metadata-only logical channels in an AVCluster.
MetadataControl	A logical object that is used to manipulate a specific metadata Property. Examples are Font Control, Color Control, etc.
MetadataControl Property	Parameter of a MetadataControl. Examples are Current, Next, Range Properties of a Font Control.
MetadataTrack	A group of MetadataChannels in a MetadataCluster that is associated with a single program.
MetadataTrack Selector	A Control that allows indicating or selecting one MetadataTrack from the MetadataCluster. The selected MetadataTrack is called the Active MetadataTrack.
Metadata Unit (DU)	Provides basic manipulation on the incoming logical MetadataChannels.
Mixer Unit (MU)	Mixes a number of logical input channels into a number of logical output channels.
MLP	Meridian Lossless Packing
MotionVector	A couple of coordinates (x,y) defining an offset from the current MacroBlock position in the current VideoFrame into the previous VideoFrame. The current MacroBlock position is defined by two coordinates (xMB, yMB) defining the position of the upper left pixel of that MacroBlock.
MPEG	Moving Pictures Expert Group.
MUD	Mixer Unit Description.
NAL	Network Abstraction Layer, See [IEC14496_10]
NAL Unit	An integer number of bytes, preceded by a ony-byte NAL Header indicating the type of data in the NAL Unit.
OTD	Output Terminal Description.
Output Pin	A logical output connection to an Entity. Carries a single AVCluster.
Output Terminal (OT)	An outlet for AV information flowing out of the AVCore.
Processing Unit (PU)	Applies a predefined process to a number of logical input channels.
PUD	Processing Unit Description.
Router Unit (RU)	Provides the means to assemble an outgoing AVCluster from multiple incoming AVClusters.
RUD	Router Unit Description.
Selector Unit (SU)	Selects from a number of input AVClusters.
ServiceInterval	A grouping of USB (micro)frames that are related.
ServiceIntervalPacket	A packet that contains all the VideoSamples that are transferred over the bus during a ServiceInterval.
SI	ServiceInterval.

Term	Description
SIP	ServiceIntervalPacket.
SIPDescriptor	A 4-byte field present at the beginning of every SIP, providing information about the layout of the SIP. Only present when transporting Extended Audio Format Type I data.
SOF	Start of Frame ([USB2.0]).
Stereo 3D Video	A video stream consisting of 2 separate video channels, providing a stereoscopic 3-dimensional video experience. One video channel is intended for the left eye, and the other video channel is intended for the right eye.
SubHeader	A header containing additional information, related to the data in the SIP.
SUD	Selector Unit Description.
Terminal	A logical object inside an AVCore that represents a connection to the AVCore's outside world.
Transfer Delimiter	A unique token that indicates an interruption in an isochronous data packet stream. Can be either a zero-length data packet or the absence of an isochronous transfer in a certain USB (micro)frame.
Unit	A logical object inside an AVCore that represents a certain AV functionality.
VideoBundle	VideoChannels are physically organized into VideoTracks, and VideoTracks are then organized into the VideoBundle where each VideoVhannel has a VideoChannel Type associated with it. Very similar to the VideoCluster concept, but the physical characteristics of the video stream, such as VideoFrame Rate are retained in the VideoBundle.
(Logical) VideoChannel	A logical transport medium for a single video channel. Makes abstraction of the physical Properties and formats of the connection. Is identified by VideoChannel Type: i.e. observation location. Examples are Channels for left eye, right eye.
VideoChannel Type	The observation location of a VideoChannel. Uniquely identifies the VideoChannel. Examples are Channels for left eye (OL001), right eye (OL002).
VideoCluster	The group of all the video-only logical channels in an AVCluster.
VideoControl	A logical object that is used to manipulate a specific video Property. Examples are Contrast Control, Zoom Control, etc.
VideoControl Property	A parameter of a VideoControl. Examples are Current, Minimum, Maximum and Resolution Properties of a Brightness Control.
VideoFrame	One of the many still images which compose a complete moving picture or VideoSequence.
VideoParticle	The basic structure used to send video data over USB. Can be one VideoSample, two VideoSamples that share Chrominance information, or individual Luminance and Chrominance components of one VideoSample.
VideoPayload	All data bytes related to one VideoFrame. Includes NAL Headers, InfoBlock Headers, etc. The AVHeader is not part of the VideoPayload.
VideoSample	The basic representation of a video signal, sampled (digitized) at a specific moment in time.
VideoSequence	A sequence of closely related VideoFrames that together make up a movie or video clip.
VideoSlot	A collection of VideoSubSlots, each containing a VideoSample of a different physical video channel, taken at the same moment in time.
VideoStream	A concatenation of a potentially very large number of VideoSlots ordered according to ascending time where the VideoSamples are formatted according to one of the VideoFrame and VideoSample Formats described in this specification and where the video channels are organized in a VideoBundle.
VideoStreamConfig (VideoStream Configuration)	An XML Description that provides all the information necessary to fully characterize a VideoStream. Includes a VideoBundle, VideoFrame, and VideoSample component.
VideoStreamConfigList	A list of VideoStreamConfig Descriptions used to indicate the different VideoStream Configurations an AVData Entity supports.
VideoSubSlot	Holds a single VideoParticle.
Viewpoint	A position from which an event (program) is observed.
WMA	Windows Media Audio.
XML Descriptions	See Descriptions.

1.5. Conventions and Notations

This specification uses an XML Document, called the AV Description Document (AVDD) to describe the full class-specific functionality of an AVFunction. For details, see Section 4.3, "AV Description Document".

A formal definition of the AVDD XML language can be found in the [AVSCHEMA] document. This specification contains references to XML elements, attributes, and type definitions. The following conventions and notations are adopted to reference those components.

- XML elements are indicated in the text by enclosing the name of the element in angle brackets (<>) and using the `courier new` font for that element name (including the brackets), such as `<volume>`.
- XML attributes to elements are indicated in the text by preceding the name of the attribute with the @ sign and enclosing the @ sign and name of the attribute in angle brackets (<>), and using the `courier new` font for that attribute name, such as `<@id>`.
- If an attribute pertaining to a specific element needs to be indicated, then the complete qualified name of the attribute is used. The qualified attribute consists of the name of the element, followed by the @ sign followed by the attibute name and enclosing the result in angle brackets (<>), and using the `courier new` font for that qualified attribute name, such as `<brightness@rw>`.
- If descendant element relationships need to be indicated, then the fully qualified descendant element name can be constructed, starting from a certain ancestor element and concatenating all intermediate descendant names, preceded by the colon (:) until the intended descendant is reached and enclosing the result in angle brackets (<>), and using the `courier new` font for that qualified descendant element name, such as `<avConfiguration:avControlInterface:alternateSetting:featureUnit:brightness>`.
- Fully qualified descendant attribute names follow the same rules, such as `<featureUnit:brightness@rw>`.
- Sometimes there is a need to use class or category names rather than specific element or attribute names. Those names are formatted as decribed above and then italicized, such as `<entityName>` or `<avControl>`.

1.5.1. Byte Ordering

All multiple byte fields in this specification are interpreted as and moved over the bus in little-endian order, i.e., LSB to MSB unless otherwise specified.

2. Management Overview

The USB is very well suited for transport of audio and video ranging from low fidelity voice connections to High Definition video and high quality, multi-channel audio streams. USB has become a ubiquitous connector on modern PC's and mobile devices 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 connector available today. With the advent of SuperSpeed USB, consumers can count on this medium to meet all of their AV needs today and into the future. Many applications from communications, to entertainment, to recording and playback, can take advantage of the audio and video features of USB.

In principle, a versatile bus specification like USB provides many ways to propagate and/or control digital media. For the industry, however, it is very important that media transport mechanisms be well defined and standardized on USB. Only in this way can interoperability be guaranteed among the many possible AV devices on USB. Standardized AV transport mechanisms also help keep software drivers as generic as possible. The AV Device Class described in this document (and some other related documents – see Section 2.1, "Specification Documents Organization") satisfies these requirements. Other Device Classes that address audio and/or video in some way should refer to this document for their AV interface specification.

An essential issue in media stream processing is synchronization of the data streams, especially for audio. Indeed, the human ear easily detects the smallest artifacts. Therefore, a robust synchronization scheme based on isochronous transfers has been developed and incorporated in the USB Specification. The AV Device Class Definition adheres to this synchronization scheme whenever AV data needs to be transported over the bus via an isochronous pipe.

Although the AV Device Class Definition is very rich and complete in features, it should be pointed out that the defined architecture is intrinsically very modular and composed of a number of optional building blocks so that it can cover a wide variety of applications, ranging from very simple devices, such as a webcam or a simple microphone up to very complex devices, such as a High Definition TV set with built-in video rendering engines that have capabilities similar to high-end video cards used in today's gaming computers. The same paradigm and Application Programming Interface (API) can be used to address all these cases (from very simple to extremely complex). The AV Device Class Definition has been designed to scale gracefully with the application.

Furthermore, to accommodate lighter-weight Controller implementations, different levels of performance and complexity have been and will be defined (through AV Profile Definitions) so that not all Controller software stacks need to implement support for the entire rich feature set of the specification. Instead, they can claim compliance to only a certain level of complexity, thereby greatly reducing the footprint of their implementation. As a consequence, however, these Controller implementations may not interoperate with all USB AV devices, but only with a subset of those.

The AV Device Class uses an XML Document, called the AV Description Document (AVDD) to convey all the class-specific information about the AVFunction, including detailed topology information, to the Controller.

However, in order to provide some level of out-of-the-box interoperability, all AV-compliant devices are also required to support a set of Legacy View Descriptors that expose a (potentially reduced) AV functionality, enough to be useful to any Controller that does not want to or is not able to parse the full AV Description Document.

Many of the features of the AV Device Class Definition take advantage of the new features provided in the USB 3.0 Specification. With the additional bandwidth made available, SuperSpeed USB operation allows the transport of multiple channels of high bit rate audio and video. This expands the range of solutions that can be provided by USB AV devices.

2.1. Specification Documents Organization

The following figure provides a visual overview of how the different documents that together comprise the USB AV Specification are organized and how they interrelate. (Rectangles represent actual documents; rounded rectangles group documents together)

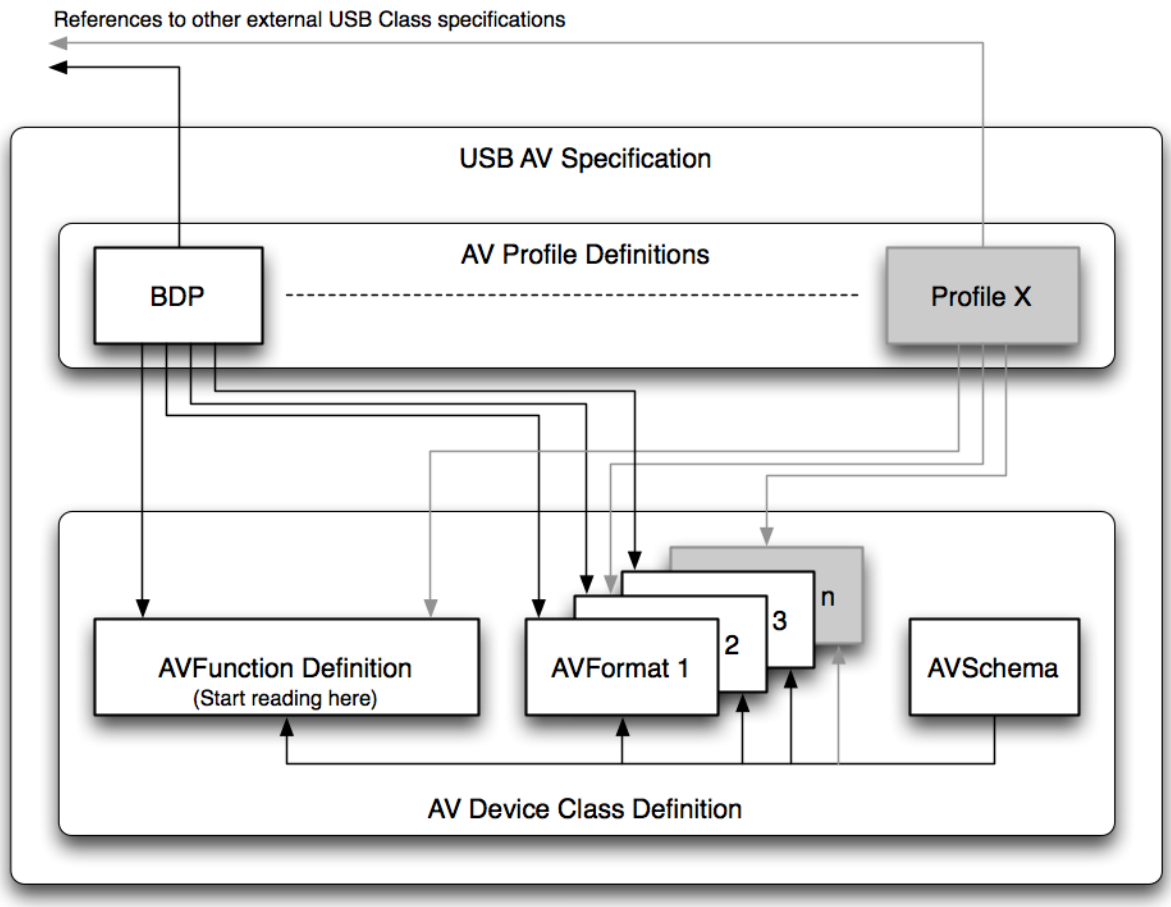


Figure 2-1: USB AV Specification Document Organization

The total set of documents related to USB AV is called the USB AV Specification. Within the USB AV Specification, there are two families of Definition documents:

- AV Device Class Definition
- AV Profile Definitions

2.1.1. AV Device Class Definition

The set of documents that comprise the AV Device Class Definition contain all necessary information for a designer to build a USB-compliant device that incorporates AV functionality. The AV Device Class Definition itself is again split into several documents:

- AV Device Class Overview & AVFunction Definition (this document), a.k.a. AVFunction Definition
- Multiple AVFormat Definitions
- AVSchema XML Schema

The AVFunction Definition provides an overview of the AV Device Class architecture and defines all of the AV concepts such as Terminals, Units, AVData Entities, etc. and explains how these different building blocks can be used together to model (almost) any AV functionality. It further specifies all class-specific Descriptions, expressed in XML in the AV Description Document, that fully describe the AV functionality available in a specific AVFunction implementation. In addition, all available Legacy View Descriptors are fully defined in this document. Finally, it explains the use of class-specific Control Sequences that allow full AVFunction control.

For each supported class of AVStream Configurations, there is an AVFormat Definition document that defines a standard way of transporting this class of audio/video over the USB. Provisions have been made so that vendor-specific AV formats and compression schemes can be handled as well.

Finally, the AVSchema document describes the syntax that is used for the AV Description Document in the form of an XML Schema.

2.1.2. AV Profile Definitions

The AV Device Class Definition provides all the 'tools' to describe and build (almost) any AVFunction on the USB. As stated before, Profile Definitions are used to narrow down the level of complexity and the actual functionality that may be used in AVFunctions that comply with a particular Profile Definition. This way, driver software can implement support only for targeted Profiles without having to provide support for all possible AVFunction variations allowed by the full AV Device Class Definition.

AV Profile Definitions therefore contain only restrictions and limitations to be imposed on the AV Device Class Definitions and they do not add additional requirements beyond those already specified in the AV Device Class Definition.

2.2. Specification Focus Areas

In this version of the USB AV Specification, not all formats, features and building blocks have been fully defined in the AV Device Class Definition. Only those formats, features and building blocks necessary to support the initial Profile Definitions have been fully specified.

Nevertheless, the first part of the AVFunction Definition (in particular, Section 4, "High-level AVFunction Overview" and Section 5, "AVFunction Structural Elements") provides a complete overview of the AV Device Class architecture and all the features and building blocks that are required to describe (almost) any AVFunction so that the reader can get a good understanding of the richness and functionality the USB AV Specification will offer in the future.

At this time, only one Profile Definition is defined:

- Basic Device Profile (BDP)

The Basic Device Profile Definition describes a fairly simple AVFunction, such as a PC monitor with one built-in screen and a pair of stereo speakers or a USB-to-HDMI dongle that would allow a cell phone to connect to a TV set. Another example would be a webcam with built-in microphone or a cell phone, acting as a USB Device, connected to the TV set, acting as the USB Host.

This specification defines the following Entities. The Entities marked in gray text are currently not fully specified in this version of the specification. In subsequent Sections, paragraphs that refer to concepts, features, or building blocks that are not fully defined in this version are also marked in gray text.

- Input Terminal
- Output Terminal
- Mixer Unit
- Metadata Unit
- Selector Unit
- Feature Unit
- Effect Unit
- Processing Unit
- Converter Unit
- Router Unit
- GraphicsEngine Entity
- AVData In Entity
- AVData Out Entity
- Encoders and Decoders

The Basic Device Profile Definition only uses the following Entity subset:

- Input Terminal
- Output Terminal
- Mixer Unit

- Feature Unit
- Converter Unit
- Router Unit
- AVData In Entity
- AVData Out Entity

The Basic Device Profile Definition uses three possible formats to move AVStreams over the USB so that at this time three Format Definitions (AVFormat 1 – Video-Only over Bulk, AVFormat 2 – Audio-Only over Isochronous, and AVFormat 3 – Video-Only over Isochronous) are necessary.

In the organizational diagram above, the shaded parts are currently not defined.

3. Fundamentals

This specification has been written with flexibility and expandability in mind. As such, no assumptions have been made as to where the different actors in the AV system are located. The following actors can be identified:

- From a USB ecosystem perspective:
  - The USB Host
  - The USB Device.
- From a USB AV perspective:
  - The Controller issues AVControl Sequences to the Controllee.
  - The Controllee responds to the AVControl Sequences received from the Controller. It also issues Notifications to the Controller.
- From a streaming perspective:
  - The Source, delivering AV content to the system
  - The Sink, consuming AV content from the system.

Although USB is naturally designed to collocate the USB Host and the System Controller into the same appliance, in some cases, it may be desirable to locate System Controller functionality in the USB Device. For example, when a portable device, such as a mobile phone, is hooked up to a TV set, the portable device may want to be in control of the system, generating the UI and controlling the flow of content. However, for charging purposes, the TV may still be the USB Host, providing V_{BUS} (and thus charging current) to the system.

In this version of the specification, it is assumed that the Controller functionality always resides on the USB Host. Likewise, it is assumed that the Controllee functionality always resides on the USB Device. Throughout the specification, the words Controller and Controllee are used rather than (USB) Host and (USB) Device in the appropriate context.

4. High-level AVFunction Overview

In many cases, AV functionality does not exist as a standalone device. It is one capability that, together with other functions, constitutes a "composite" device. An example of this is a TV set, which can incorporate audio and video, data storage, and transport control. The AV Functionality, called the AVFunction in this specification, is thus located at the interface level in the device class hierarchy. From a USB perspective, the AVFunction is exposed via a number of associated interfaces that together provide access to its capabilities. The Controller can manipulate the internals of the AVFunction and send and receive various AVStreams through this association of interfaces.

The following figure presents a high-level visual overview of the AVFunction architecture and its components.

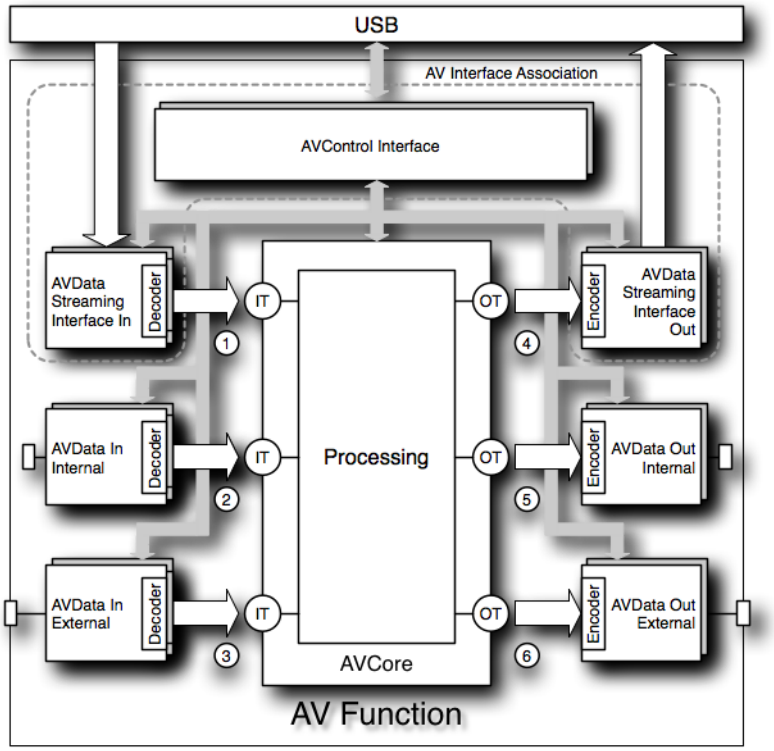


Figure 4-1: AVFunction Global View

4.1. AVFunction

The AVFunction can be divided into two major zones, inside the AVCore where most of the AV-related control takes place and the rest of the AVFunction outside the AVCore that is primarily dealing with interfacing with the outside world, bringing AVStreams into and out of the AVFunction.

4.1.1. Inside the AVCore

At the center of the AVFunction is the AVCore that contains most of the AV processing functionality. Inside the AVCore, AV functionality is described through an *abstract logical model*, based on a small set of predefined objects, called Units. Each Unit provides a certain AV sub-functionality. Each Unit exposes a number of AVControls that allow the Controller to influence and manipulate the AV processing offered by the Unit. By combining a number of different Units together, (almost) any AV functionality can be modeled efficiently.

In addition to Units as basic building blocks, this specification also supports a GraphicsEngine Entity to model an optional graphics processing subsystem that can be controlled via a graphics command language, such as the OpenGL/ES (see [KHRONOS]).

The AVCore can be considered a 'closed box' that has very distinct and well-defined points of access. These points of access are called Input Terminals (IT) and Output Terminals (OT) (see Figure 4-1).

Within the AVCore, the AV data that flow over the connections between Units and Terminals are also considered to be *logical* concepts. Such a flow of AV data is called an AVCluster, and complete abstraction is made of the actual physical characteristics of the AV data. An AVCluster can have a VideoCluster, AudioCluster, and MetadataCluster component. The number of logical channels and for each logical channel, the Channel Type, fully defines each Cluster. Physical characteristics, such as AudioFrame Rate (sampling frequency) and bit resolution for audio or VideoFrame Rate and pixel resolution for video are considered irrelevant and therefore not used.

It is important to understand that this specification does not aim at describing the underlying hardware and/or software that is used to realize the AV functionality. In particular, the exposed topology, describing how the building blocks (Units) are interconnected, does not necessarily (have to) reflect the actual physical location of certain controls and processes as they are implemented in hardware and/or software. For example, a Volume Control in a Feature Unit (see Section 6.6, "Feature Unit") may be located somewhere in between an Input Terminal and an Output Terminal in the logical model whereas the actual volume control mechanism is most likely going to be implemented in hardware right before the power amplifiers that drive the speakers (in the analog domain).

4.1.2. Outside the AVCore

Outside the AVCore, AVStreams enter or leave the AVFunction using the concept of an AVData Entity. The AVData Entity describes the physical characteristics of the AVStream it can process. An AVData Entity can represent either a source of AV content (AVData In Entity) or a sink for AV content (AVData Out Entity) from the perspective of the AVCore.

An AVData In Entity receives the AVStream – either from the USB or from an internal or external source – and applies some form of decoding on the received AVStream. The decoding process can be as trivial as simply de-interleaving a stereo AudioStream into Front Left and Front Right AudioChannels up to fully decoding an MPEG4 AVStream into all of its individual components (VideoChannels, AudioChannels, and MetadataChannels). After decoding, the data eventually enters the internals of the AVCore through the associated Input Terminal (interfaces 1, 2 and 3 in Figure 4-1) and 'loses' its physical characteristics.

Likewise, there is always some sort of encoding process taking place in the AVData Out Entity after the data leaves the AVCore through an Output Terminal, regaining its physical characteristics. The encoding process again can range from rather trivial to very complex. After encoding, the data is delivered either over USB or to an internal or external sink. (interfaces 4, 5, and 6 in Figure 4-1).

As stated before, AVStreams do not necessarily have to travel over USB. AVStreams that originate from an internal source (such as a built-in TV tuner, for example) or from external input connectivity (such as a set of Component & Audio input connectors) can be modeled using the AVData In Entity – Input Terminal concept (interfaces 2 and 3 in Figure 4-1, respectively). Likewise, AVStreams that are destined for an internal sink (such as a built-in screen and speakers) or for external output connectivity (such as an HDMI output connector) can be modeled using the Output Terminal – AVData Out Entity concept (interfaces 5 and 6 in Figure 4-1, respectively). AVData Streaming Interfaces are a type of AVData Entities where the AVStreams are delivered to or from the AVCore over the USB.

4.2. AVFunction Interfaces

AVFunctions are addressed via USB through their AV interfaces. Each AVFunction shall have a single AVControl Interface and can have zero or more AVData Streaming-Out interfaces and zero or more AVData Streaming-In interfaces.

The AVControl Interface is primarily used by the Controller to access the AVControls that reside within the AVFunction. It is also used to stream video content to and from the AVFunction via appropriate AVData Entities. It can also be used to access, control, and provide information (like textures, graphics commands, etc.) to an optional graphics subsystem within the AVCore.

AVData Streaming Interfaces are used to transport AVStreams into and out of the AVFunction over the USB. Every AVData Streaming-In interface contains a single isochronous Data OUT endpoint (interface 1 in Figure 4-1). Every AVData Streaming-Out interface contains a single isochronous Data IN endpoint (interface 4 in Figure 4-1). In some cases, a feedback isochronous endpoint may also be present for synchronization purposes.

Note: In this specification, AVData Streaming Interfaces (and more in general, AVData Entities) use a naming convention (In vs. Out) where the AVCore is the point of reference. AVData In Entities provide input to the

AVCore; AVData Out Entities consume output from the AVCore. This is different from the naming convention used for USB interfaces and endpoints where the Controller (Host) is the point of reference: An AVData Streaming-*In* interface has a USB Data *OUT* endpoint and an AVData Streaming-*Out* interface has a USB Data *IN* endpoint.

All functionality pertaining to controlling parameters that directly influence video, audio and metadata perception (like volume, brightness, font size, etc.) are located inside the central rectangle, the AVCore. (For a more detailed description of the internal architecture of the AVCore, refer to Section 5, "AVFunction Structural Elements".) Streaming-related and codec-related AVControls are located in the AVData Entities.

All AVControls present in the entire AVFunction (both inside and outside the AVCore) are addressed and controlled through the single AVControl Interface. This means that all Control Sequences (see Section 9, "Requests and Control Sequences") are always directed to the AVControl Interface of the AV function. The Control Sequence contains enough addressing information to uniquely identify the targeted AVControl.

The AVControl Interface uses the default control endpoint 0 for all standard USB interface requests. In addition, a pair of bulk endpoints (one IN, one OUT) within the AVControl Interface is used as the main communication channel for all class-specific control information. This pair of bulk endpoints is called the Control Bulk Pair (CBP).

The AVData Streaming Interfaces also use the default control endpoint 0 for all standard USB interface requests. However, all class-specific control information for the AVData Streaming Interfaces is conveyed through the CBP, located in the AVControl Interface.

4.3. AV Description Document

In general, the AV Device Class Definition does not use the usual class-specific USB descriptors to convey class-specific information about the AVFunction to the Controller (except for Legacy View). Instead, all functional Entities within the AVFunction have a class-specific Entity Description associated with them, expressed in XML (see Section 8, "AV Descriptors and the AV Description Document"). The ensemble of Unit, Terminal, and AVData Entity Descriptions provides a full functional description of the AVFunction to the Controller. This information is retrieved from the device as a single XML Document, called the AV Description Document (AVDD), typically at enumeration time. By parsing this XML Document, a generic AV driver should be able to fully control the AVFunction.

4.4. Legacy View Descriptors

To allow Controllers to interoperate (most likely, at a reduced functionality level) with the AVFunction when the Controller does not have a full-featured AV Class driver available (during the boot process, for example), the AVFunction shall provide a set of Legacy View Descriptors that contain enough information for the Controller to use the AVFunction to a certain operational level without having to parse the AVDD. For details, refer to Section 8.3, "Legacy View Descriptors".

5. AVFunction Structural Elements

The following sections describe the building blocks that are used by this specification to create an abstract logical model that allows describing (almost) any AV functionality that can exist as part of a USB device.

5.1. Entities

To be able to manipulate the physical Properties of an AV Function, its functionality is divided into manageable, addressable Entities. This specification defines different types of Entities as illustrated in the following figure:

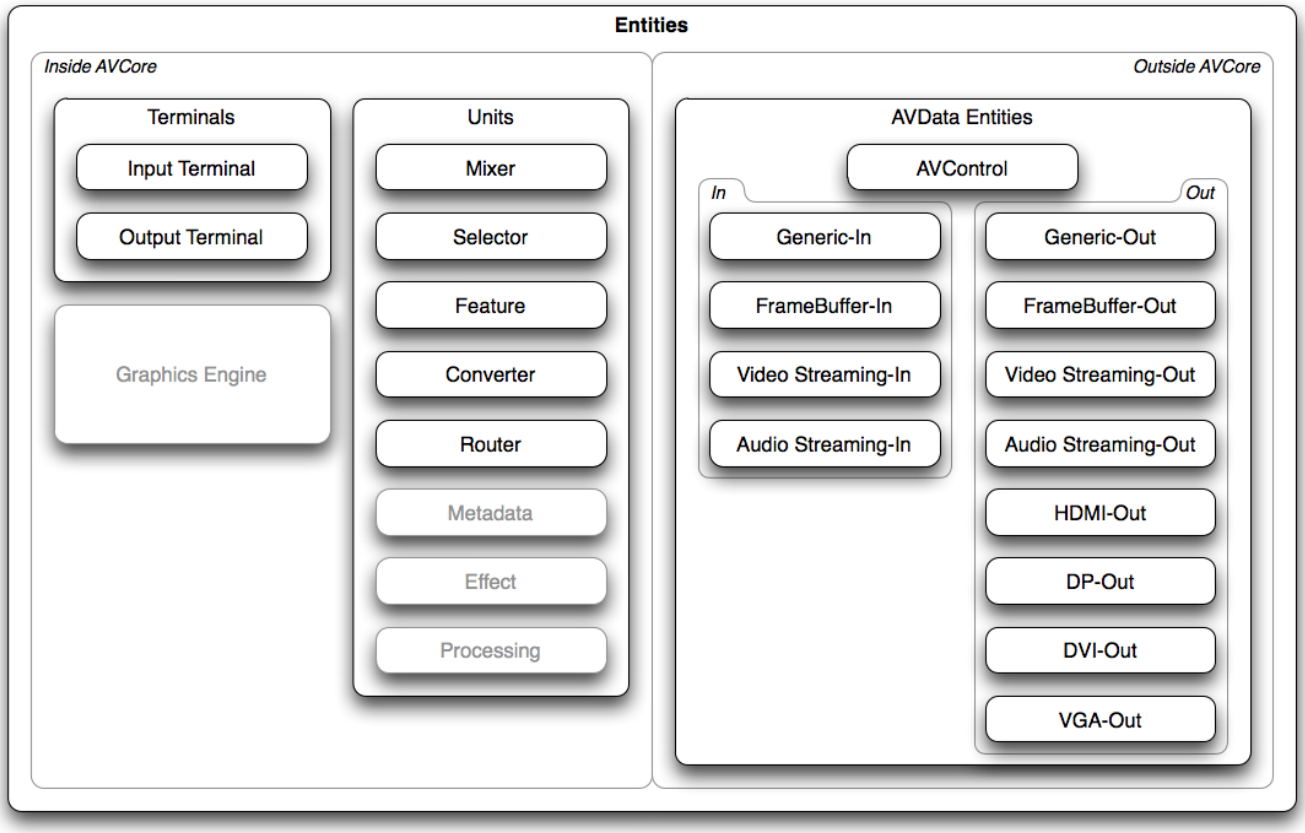


Figure 5-1: Entity Hierarchy

Inside the AVCore:

- Units – these are the basic building blocks that are used to describe the AV functionality within the AVCore. See Section 5.1.1, "Units".
- Terminals – these are the 'ports' through which logical AVClusters enter and leave the AVCore. See Section 5.1.2, "Terminals".
- GraphicsEngine Entity – this is used to expose a graphics subsystem, embedded in the AVFunction that can interact with the various AVClusters in the AVCore and generate a number of output AVClusters. The graphics subsystem is controlled using a graphics command language, such as OpenGL/ES. There can only be one GraphicsEngine Entity in an AVFunction.See Section 5.1.4, "GraphicsEngine Entity".

Outside the AVCore:

- AVData Entities – these are used to describe and manipulate the AVControls that are related to the physical streaming aspects of an input source or an output sink to the AVCore. In addition, they are used to describe and manipulate the decoding process that takes place just before an AVStream enters the internals of the AVCore as an

AVCluster. Likewise, they are used to describe and manipulate the encoding process that takes place just after an AVCluster leaves the internals of the AVCore as an AVStream. These Entities are always associated with a particular Terminal. Lastly, AVData Entities can describe and manipulate certain aspects of their affiliated Clock Domain. See Section 5.1.5, "AVData Entities" for more details on AVData Entities.

Note AVData Streaming Interfaces are classified as a type of AVData Entity. However, since they are standard USB interfaces, to interact with some of their functionality, they use standard USB requests rather than the class-specific Control Sequences used by other AVData Entities to control the same functionality.

Each Entity shall have an EntityID that shall be unique within the AVFunction. This ID is used as the primary component of the address of a particular AVControl within the AVFunction. The maximum number of Entities inside an AVFunction can be up to 65,535. EntityID zero (0x0000) is reserved and shall never be used for an addressable Entity inside the AVFunction since it is used in managing the transport of CBP messages (see Section 9.3.4, "Null Message"). The mandatory AVControl Interface shall always have its EntityID set to one (0x0001) so that Controller software can access the AVControls inside the AVControl Interface Entity at all times.

In addition, each addressable object (Entity, AVControl, etc.) can have an optional ReferenceID (`<entityName@refID>`) that can be used to uniquely reference the Entity outside the AVFunction. For example, an implementation may choose to provide a Globally Unique Identifier (GUID) for some of its Entities so that a binding can be created between these Entities and an HID interface in the composite device that may be used to control certain aspects of the Entity that are outside the realm of this specification. As an example, consider an AVData Entity that represents a camera where certain aspects of the camera (zoom, panning, etc.) are controlled through an associated HID interface.

5.1.1. Units

Units provide the basic building blocks to fully describe the AV functionality within the AVCore of most AVFunctions. Units can only be part of the AVCore. Each Unit contains a logical grouping of addressable Audio, Video, and Metadata Controls that together constitute a certain self-contained AV sub-functionality with a predefined internal topology. For example, a Mixer Unit accepts multiple input AVClusters and contains a number of Mixer Controls that allow the incoming AVChannels to be mixed together into a single output AVCluster.

It is assumed that the different Unit types defined in this specification are adequate to model (almost) any AVCore. The desired overall AV functionality can always be broken down into smaller functional blocks, each of which can be modeled by one of the predefined Unit types. By connecting the necessary Units via their Input and Output Pins according to the required topology, the entire AVCore can be modeled.

A Unit has one or more Input Pins and a single Output Pin, where each Pin carries an AVCluster of zero or more logical VideoChannels (the VideoCluster), zero or more logical AudioChannels (the AudioCluster), and zero or more logical MetadataChannels (the MetadataCluster) inside the AVCore (see Section 5.4, "AVCluster" for more information on AVClusters and their VideoCluster, AudioCluster, and MetadataCluster components). Input Pins of a Unit are numbered starting from one up to the total number of Input Pins on the Unit. The single Output Pin number is always one.

5.1.2. Terminals

A Terminal is a logical representation inside the AVCore of a physical component of the AVFunction that either delivers AVStream to the AVCore or consumes processed AVStream from the AVCore. Two types of Terminals are defined:

- An Input Terminal (IT) represents a logical starting point for an AVCluster inside the AVCore. It represents the AVData In Entity in the AVCore and delivers content as a logical concept, the AVCluster (see 5.4, "AVCluster") via its Output Pin.
- An Output Terminal (OT) represents a logical ending point for an AVCluster in the AVCore. It represents the AVData Out Entity in the AVCore and accepts processed content as a logical concept, the AVCluster, (see 5.4, "AVCluster") via its Input Pin and delivers that content to its associated AVData Out Entity.

From the AVCore's perspective, the most obvious example of what physical component a Terminal can represent is an AVData Streaming Interface with its embedded USB endpoint. An AVStream flowing into the USB OUT endpoint of an

AVData Streaming-In Interface eventually enters the AVCore as an AVCluster through the Input Terminal that represents that AVData Streaming Interface. Likewise, a processed AVCluster leaves the AVCore through the Output Terminal as an AVStream. The Output Terminal represents the AudioStreaming-Out Interface with its embedded USB IN endpoint through which the AVStream is eventually delivered to the Controller.

Another example could be a built-in Digital to Analog converter, modeled as an AVSink Entity that is represented as an Output Terminal in the AVCore's model.

5.1.3. Combining Units and Terminals

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.

Units and Terminals are wired together by connecting their Input and Output Pins according to a certain topology to realize the desired overall functionality. Note that it is perfectly legal to connect the Output Pin of an Entity to multiple Input Pins of different Entities (or even the same Entity), effectively creating a one-to-many connection. Connecting multiple Output Pins together is not allowed.

The information, traveling over Input and Output Pins is not necessarily of a digital nature. It is perfectly possible to use the Terminal and Unit model to describe fully analog or even hybrid AVFunctions.

Important Note: The fact that Input and Output Pins are connected together indicates that the protocol and format, used over these connections (analog or digital), is compatible on both ends.

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

This specification describes the following different types of standard Units and Terminals that are considered adequate to represent most AVFunctions available today and in the near future:

- Input Terminal (IT). See Section 6.1.
- Output Terminal (OT). See Section 6.2.
- Mixer Unit (MU). See Section 6.3.
- Metadata Unit (DU). See Section 6.4.
- Selector Unit (SU). See Section 6.5.
- Feature Unit (FU). See Section 6.6.
- Effect Unit (EU). See Section 6.7.
- Processing Unit (PU). See Section 6.8.
- Converter Unit (CU). See Section 6.9.
- Router Unit (RU). See Section 6.10.

5.1.4. GraphicsEngine Entity

The GraphicsEngine (GE) Entity is introduced to allow an embedded graphics subsystem to be exposed within the AVCore. There can only be one GE Entity present inside an AVCore. If there are multiple graphics processors available within the graphics subsystem, they are considered to be part of the same GE Entity from the AVCore's perspective. The AVCore provides a bidirectional conduit for any control and other information, such as textures etc. that needs to be exchanged between the Controller and the GE Entity. This control and information exchange happens over the AVControl Interface CBP and is implemented using the OpenGL/ES graphics command language.

Although the graphics subsystem is in many respects completely independent of the rest to the AVCore, some interaction with the rest of the AVCore may be desirable. Therefore, the GE Entity has one or more Input Pins and one or more Output Pins, where each Input Pin carries an AVCluster of zero or more logical VideoChannels (the VideoCluster), zero or more logical AudioChannels (the AudioCluster), and zero or more logical MetadataChannels (the MetadataCluster). The GE Entity discards the audio and MetadataCluster since it operates only on video. Input Pins of the GE Entity are numbered starting from one up to the total number of Input Pins on the GraphicsEngine Entity.

Since the graphics subsystem is potentially capable of generating multiple video output streams simultaneously, the GE Entity provides multiple Output Pins to deliver those multiple output streams to the AVCore for further processing. Each Output Pin carries an AVCluster of one or more logical VideoChannels. No AudioClusters or MetadataClusters are present in the outgoing AVClusters (the GE Entity only deals with video). The Output Pins are also numbered from one up to the total number of Output Pins on the GE Entity.

For more details on the GE Entity, see Section 6.11, "GraphicsEngine Entity".

5.1.5. AVData Entities

AVData Entities describe and manipulate certain aspects of the AV sources (AVData In) and sinks (AVData Out) that may be present in the AVFunction. They also describe and manipulate the behavior of the decoding and encoding processes that take place whenever an AVStream enters the AVCore (decoder) or leaves the AVCore (encoder). AVData Entities can represent sources and sinks, internal to the AV Function, such as a built-in TV tuner, or a built-in display. They can also represent sources and sinks external to the AV Function, such as a display, connected to the AVFunction via an HDMI cable, or a media player running on a PC sourcing AV content to the AVFunction over the USB via an AVData Streaming Interface.

AVData Entities live outside the AVCore and are dealing with the more physical aspects of the AVStreams that are entering and leaving the AVFunction. Therefore, it is important that the physical aspects of the AVStreams a particular AVData Entity supports are accurately advertised so that Controller software can select appropriate settings on the AVData Entity to match the characteristics of the AVstreams the Controller wants to exchange with those AVData Entities.

Note: This only applies to those types of AVData Entities that can actually exchange AVStreams with the Controller; i.e. AVData FrameBuffer Entities (see Sections 6.12.1.2 and 6.12.2.2) and AVData Streaming Interfaces (see Sections 6.12.1.3, 6.12.1.4, 6.12.2.4, and 6.12.2.5).

To accurately describe the timing aspects related to the operation of an AVData Entity, the concept of a Clock Domain is introduced (see Section 7.9, "Clock Model"). Every AVData Entity is affiliated to a Clock Domain, either implicitly or explicitly. An AVData Entity is either a Clock Domain Source (the AVData Entity delivers the Clock Domain Reference Clock to the Domain) or a Clock Domain Sink (the AVData Entity consumes the Clock Domain Reference Clock. AVData Generic Entities (see Sections 6.12.1.1 and 6.12.2.1) and AVData FrameBuffer Entities can optionally explicitly indicate their affiliated Clock Domain and how they are related to that Clock Domain (when this information is useful to the Controller to properly operate the AVFunction). AVData Streaming Interfaces are required to explicitly indicate their Clock Domain affiliation because for isochronous interfaces, timing aspects and relationships are important.

Each AVData Entity has an associated AVData Entity Description (AED). The AVData Entity Description contains all necessary information (in XML format) to fully identify and describe the Entity, the AVStream Configurations it supports, and the Clock Domain to which it is affiliated.

The Descriptions for all Entities are further detailed in Section 8, "AV Descriptors and the AV Description Document" of this document.

For more details on AVData Entities, see Section 6.12, "AVData Entity".

5.2. AVControls

Inside an Entity, functionality is further described through AVControls. These can be Video, Audio, Metadata, Decoder, and Encoder Controls. A VideoControl provides access to a specific Video Property (Brightness, Contrast, etc.). An AudioControl provides access to a specific Audio Property (Volume, Bass, etc.). A MetadataControl provides access to a specific Metadata Property (Font Size, Font Color, etc.). Likewise, a Decoder Control or Encoder Control provides access to a Decoder Property (mode, scaling, etc.) or Encoder Property (bit rate, quality, etc.) respectively.

Each Entity can only contain a specific set of AVControls, permitted for use by that Entity.

It is important to note that all interactions with the AVFunction are performed via AVControls. All accessible and/or controllable parameters inside the AVFunction are modeled using the concept of the AVControl and its associated Properties.

Each AVControl has a set of Properties that can be manipulated or that presents additional information about the behavior of the AVControl. An AVControl can have the following Properties:

- Current setting Property (CUR)
- Next setting Property (NEXT)
- Ranges Property (`<ranges>`)

Properties can be Read-Only, Read-Write and even Write-Only. For details, see Section 9.4, “Control Properties”.

A Property of an AVControl provides the finest level of addressable control granularity within the AVFunction. In order to gain access to a particular AVControl, the Controller shall in general provide the following information:

- The unique ID of the Entity in which the AVControl resides.
- The Property of the targeted AVControl.
- The Control Selector (CS) indicating which type of AVControl is targeted.
- The Input Pin Number (IPN), the Input Channel Number (ICN), and the Output Channel Number (OCN) of the targeted AVControl, if applicable. (See Section 5.6, “AVControl Hierarchical Accessing Scheme” for details.)

For more details on addressing AVControls, see Section 9, “Requests and Control Sequences”.

5.2.1. Master AVControls

Video and AudioControls are often implemented on a per-channel basis. However, in order to control the settings of a particular type of AVControl, the concept of a Master Control has been defined which controls all the VideoChannels, AudioChannels, or MetadataChannels simultaneously. Note that a Master Control shall always be implemented separate from and independent of the per-channel AVControls. Changing the setting of a Master Control does not affect the settings of any of the individual per-channel AVControls. The following figure illustrates the concept.

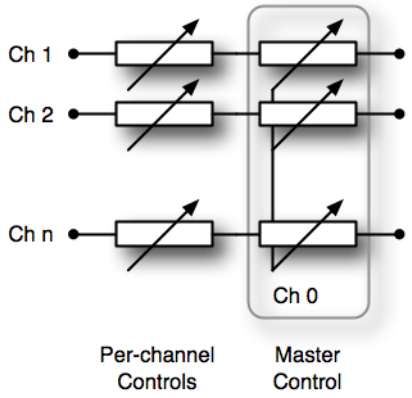


Figure 5-2: N-channel AVControl with Master Control

5.2.2. Synchronizing Multiple AVControls

In some cases, multiple AVControls need to be manipulated simultaneously in order to achieve a desired effect or to avoid unwanted transitory side effects. To accomplish this, every AVControl can have the Next setting Property that is used to preprogram an AVControl to a certain value. Only when a Commit Control Sequence is issued (to the AVFunction as a whole) will all AVControls in the AVFunction take on as their Current setting the value that was preprogrammed in their Next setting Property. This effectively provides a means to have a number of AVControls (including AVControls present in the AVData Streaming Interfaces and all other Entities) take on a new value at the same point in time. Note further that any AVControls in the entire AVFunction can be synchronized using this technique by properly interleaving sets of SET/SETR(NEXT) Control Sequences with Commit Control Sequences.

For more detailed information on Control Sequences and Property manipulation, see Section 9, “Requests and Control Sequences”.

5.3. AV Control Sequences

To access the AVControls an AVFunction exposes, the Controller uses the AV Control Sequence.

A Controller-initiated Control Sequence consists of two phases:

- In the first phase, the Controller sends a Command Message to the Controllee. The Command Message contains all the necessary addressing information for the Controllee to determine which Property of a particular AVControl is targeted and also, in case of a write operation, the new desired value(s) for the AVControl's targeted Property.
- In a second phase, after decoding the received Command Message, the Controllee executes the command and returns an appropriate Response Message, either indicating a successful completion of the command or it detects an error and returns an Error Response Message. When the command executed successfully, the Response Message may also contain the value of the AVControl's targeted Property (in the case of a read operation or a write operation with return).

There are three Controller-initiated Control Sequences defined:

- Get Control Sequence (GET): This Sequence is used by the Controller to retrieve the value of a Property of an AVControl from the Controllee.
- Set with Return Control Sequence (SETR): This Sequence is used by the Controller to set the value of a Property of an AVControl in the Controllee and, in the same atomic operation, retrieve the value of that Property, potentially after it has been rounded and/or adjusted by the AVFunction in the Controllee to match the supported resolution and minimum and maximum values for that AVControl's Property.
- Set without Return Control Sequence (SET): This Sequence is used by the Controller to set the value of a Property of an AVControl in the Controllee without retrieving the rounded or adjusted value. This is useful for those cases where the value of the Property may be rather large (such as the partial update of a SourceData Control) or where the value of the Property is used and discarded and not available for retrieval.

Not all Control Sequences are Controller-initiated. Whenever a Property value of an AVControl changes and that change is not a direct result of a Control Sequence issued to that AVControl, the AVControl sends a single-phase Notify Control Sequence to the Controller, indicating that the AVControl's Property value has changed. The new Property value is included in the Notify Control Message.

Finally, a special Null Message (NULL) is defined. It is used to indicate the end of a series of Messages.

For details on Control Sequences and how they are used to access AVControl Properties, see Section 9, "Requests and Control Sequences".

5.4. AVCluster

In general, an AVCluster is a collection of VideoChannels, AudioChannels and MetadataChannels that carry tightly related and synchronous AV information.

AVClusters are used inside the AVCore to describe the information that travels over the connections among Terminals and Units. (They have no meaning in the context of the actual data transport over the USB.) The VideoChannels AudioChannels and MetadataChannels in an AVCluster are considered to be logical channels and all the physical properties of the channels (video frame rate, video pixel resolution, audio sampling rate, audio bit resolution, etc.) are not specified and considered irrelevant in the context of the AVCore as seen through the AVControl Interface.

Important Note: The fact that an Input Pin and an Output Pin are connected together in the AVCore's topology indicates that the information flowing over the connection is compatible with both Entities that are connected.

An AVCluster is hierarchically structured as follows:

- An optional VideoCluster, consisting of:
  - One or more VideoTracks, each consisting of:
    - One or more VideoChannels
- An optional AudioCluster, consisting of:
  - One or more AudioTracks, each consisting of:
    - One or more AudioChannels
- An optional MetadataCluster, consisting of:
  - One or more MetadataTracks, each consisting of:
    - One or more MetadataChannels
- Mutually exclusive with all other types of Cluster, a RawDataCluster to pass through an opaque data stream with unknown characteristics. (See Section 5.4.4, "Raw Data" for more details.)
  - No Track concept
    - No channel concept

An implementation shall at least support one of the optional components.

The following figure illustrates the concept:

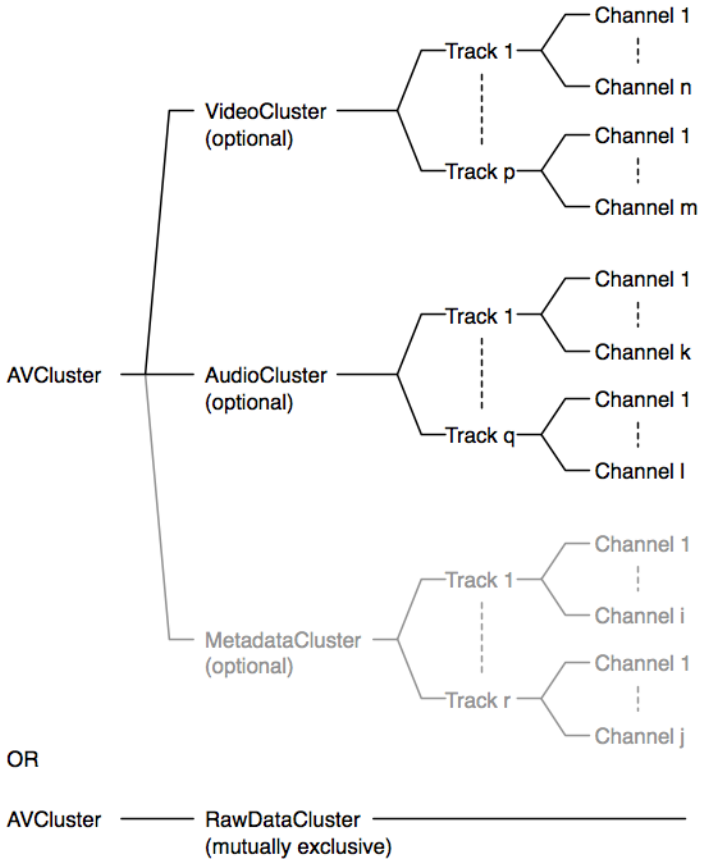


Figure 5-3: AVCluster Hierarchy

A VideoCluster can only be multi-channel if the embedded VideoChannels are really part of the same program and are there to either create a 3D experience (a single VideoTrack with multiple VideoChannels in the VideoTrack) or to provide multiple viewpoints for the same program (multiple VideoTracks in the VideoCluster, either single channel (2D) or multi-channel (3D2 or 3Dn)). Moreover, if there is also audio and metadata content in the AVCluster, then the video content in the VideoCluster shall be tightly related to that audio and metadata content from a content perspective.

Likewise, an AudioCluster can only contain multiple AudioTracks if the audio content on those different (most likely multi-channel) AudioTracks is really part of the same program. Moreover, if there is also video and metadata content in the AVCluster, then the audio content in the AudioCluster shall be tightly related to that video and metadata content from a content perspective.

Additionally, a MetadataCluster can only contain multiple MetadataTracks if the content on those different MetadataTracks is really part of the same program. Moreover, if there is also video and audio content in the AVCluster, then the metadata content in the MetadataCluster shall be tightly related to that video and audio content in the AVCluster from a content perspective.

An example could be a movie that has one VideoTrack and several different AudioTracks, each with dialogs in a different language but identical otherwise. In addition, the movie could have multiple MetadataTracks (each containing subtitles and other related textual information, such as director's notes, for example), each in a different language.

Tracks are identified using a numbering scheme in the AVCluster that starts with Track one up to the total number of Tracks in the AVCluster. This applies to VideoTracks, AudioTracks, and MetadataTracks independently.

VideoTracks in the VideoCluster are identified using a numbering scheme starting from one up to the total number of VideoTracks in the VideoCluster. The maximum number of VideoTracks in a VideoCluster can be up to 65,535. VideoChannels in a VideoTrack are identified using the label corresponding to their VideoChannel Type (see Section 5.4.1, "Multi-channel Video" below). The maximum number of VideoChannels in a VideoTrack of an AVCluster can be up to 65,535 (Channel zero is used to reference the master channel).

Likewise, AudioTracks in the AudioCluster are identified using a numbering scheme starting from one up to the total number of AudioTracks in the AudioCluster. The maximum number of AudioTracks in an AudioCluster can be up to 65,535. AudioChannels in an AudioTrack are identified using the label corresponding to their AudioChannel Type (see Section 5.4.2, "Multi-channel Audio" below). The maximum number of AudioChannels in an AudioTrack is also limited to 65,535 (channel zero is used to reference the master channel).

Finally, MetadataTracks in the MetadataCluster are identified using a numbering scheme starting from one up to the total number of MetadataTracks in the MetadataCluster. The maximum number of MetadataTracks in a MetadataCluster can be up to 65,535. MetadataChannels in a MetadataTrack are identified using the label corresponding to their MetadataChannel Type (see Section 5.4.3, "Multi-channel Metadata" below). The maximum number of MetadataChannels in a MetadataTrack can be up to 65,535 (channel zero is used to reference the master channel).

5.4.1. Multi-channel Video

For multi-channel video, several different cases need to be considered. The (basic) video information can be contained in a number of synchronous VideoChannels:

- One VideoChannel is needed for the standard 2D case where both eyes observe the same video information. This situation is referred to as 2D in this specification.
- Two VideoChannels are needed for the stereoscopic 3D case where one VideoChannel is intended for the left eye and the other is intended for the right eye. This situation is referred to as 3D2 in this specification.
- Multiple VideoChannels are needed for multiscopic systems where these VideoChannels are rendered in such a way that, depending on the observation angle, a different pair of VideoChannels (out of the total set of VideoChannels) is observable by the left and right eye. (This type of rendering systems usually involves a lenticular system placed in front of the actual display so that different VideoChannels are only visible under certain angles.) This situation is referred to as 3Dn in this specification.

This ensemble of (basic) VideoChannels is grouped into a single VideoTrack.

To support multiple viewpoints, several VideoTracks may be grouped inside a single VideoCluster.

Each VideoChannel in a VideoTrack has an associated VideoChannel Type, which shall be unique for each VideoChannel. The VideoChannel Type fully identifies the VideoChannel. In many cases, the VideoChannel Type is directly related to a certain observation location in the observation space.

This specification defines an ordered list of 64 predefined VideoChannel Types (observation locations):

Table 5-1: Predefined VideoChannel Types

Ordinal	Acronym	Description
1	**OL001**	First Observation Location (Leftmost location)
2	**OL002**	Second Observation Location
3	**OL003**	Third Observation Location
…	…	…
64	**OL064**	Last Observation Location (Rightmost location)

The labels of the form "Olxyz" in this context where xyz is a 3-digit number and the numbers follow a contiguously ascending numbering scheme from 001 to 064 are defined and reserved by this specification. OL001 is used in the 2D case (mono video). Left and Right Observation Locations (OL001 and OL002) are used in the 3D2 case (stereoscopic and autostereoscopic video) and Observation Locations OL001 through OL064 can be used in the 3Dn case (automultiscopic video). Furthermore, Observation Location OL001 corresponds to the leftmost observation location (from the viewer's perspective) and Olnnn corresponds to the rightmost observation location where nnn stands for the highest number used by a specific implementation.

This specification currently assumes that all observation locations are situated in a horizontal plane perpendicular to the rendering or generation surface.

Besides the predefined VideoChannel Types, this specification allows for custom-defined VideoChannel Types. The AV Description Document shall then contain an ordered list of custom-defined type values.

The number of VideoChannels may be different per VideoTrack.

All VideoChannels in the same VideoTrack shall have the same aspect ratio. Different VideoTracks may have different aspect ratios.

To be able to describe more complex cases in a manageable fashion, this specification imposes some limitations and restrictions on the ordering of VideoChannels in a VideoTrack and also in the VideoCluster:

- If there are VideoChannels present in a VideoTrack that correspond to some of the predefined VideoChannel Types (observation locations), then they shall appear in the order specified in the ordered list of predefined VideoChannel Types. Refer to Table 5-1, "Predefined VideoChannel Types". Any VideoChannels that correspond to custom-defined VideoChannel Types shall immediately follow the VideoChannels with predefined VideoChannel Types and they shall appear in the order as specified in the list of custom-defined VideoChannel Types in the AV Description Document (see Section 4.3, "AV Description Document").
- This ordering automatically assigns a channel number to each VideoChannel in the VideoTrack: the first VideoChannel is channel 1, the second VideoChannel is channel 2, and so on up to the last channel in the VideoTrack.
- The default VideoTrack (main viewpoint) shall come first in the VideoCluster and shall be VideoTrack one.
- Additional VideoTracks, if present, follow the default VideoTrack using a contiguously ascending numbering scheme.

5.4.1.1. Multi-channel Video Example

Suppose an AVCluster contains 2 different viewpoints for the same program content; the first main viewpoint consists of 7-channel 3Dn content, and the second viewpoint consists of only 3D2 content. Then the total number of VideoChannels in the VideoCluster will be 9 and, when entering the AVCore (through an Output Pin of an Input Terminal), the VideoChannels will be ordered as follows:

- The first seven VideoChannels (numbered from one to seven) constitute the first VideoTrack (VideoTrack 1) that contains the 7-channel 3Dn content from the main viewpoint (as intended by the content creator) where the first VideoChannel contains the content from the leftmost observation location (OL001), and the last VideoChannel contains the content from the rightmost observation location (OL007).
- The next 2 VideoChannels (numbered from one to two) constitute the second VideoTrack (VideoTrack 2) that contains the 2-channel 3D2 content from the secondary viewpoint (as intended by the content creator) where the first VideoChannel contains the content from the left observation location (OL001) and the second VideoChannel contains the content from the right observation location (OL002).

5.4.2. Multi-channel Audio

Similar to the multiple viewpoints for video, multiple multi-channel AudioTracks may be present simultaneously in the AudioCluster.

Each AudioChannel in an AudioTrack has an associated AudioChannel Type, which shall be unique for each AudioChannel. The AudioChannel Type fully identifies the AudioChannel. In many cases, the AudioChannel Type is directly related to a certain location in the listening space. A trivial example of this is an AudioTrack that contains Left and Right AudioChannels. However, multi-channel audio usually involves more elaborate situations where multiple different spatial locations are present.

This specification defines an ordered list of 51 predefined AudioChannel Types (spatial locations):

Table 5-2: Predefined AudioChannel Types

Ordinal	Acronym	Description
1	**FL**	Front Left
2	**FR**	Front Right
3	**FC**	Front Center
4	**SL**	Side Left
5	**SR**	Side Right
6	**BL**	Back Left
7	**BR**	Back Right
8	**BC**	Back Center
9	**FLC**	Front Left of Center
0	**FRC**	Front Right of Center
11	**BLC**	Back Left of Center
12	**BRC**	Back Right of Center
13	**LFE**	Low Frequency Effects
14	**LLFE**	Left Low Frequency Effects
15	**RLFE**	Right Low Frequency Effects
16	**TC**	Top Center
17	**TFL**	Top Front Left
18	**TFR**	Top Front Right
19	**TFC**	Top Front Center
20	**TSL**	Top Side Left
21	**TSR**	Top Side Right
22	**TBL**	Top Back Left
23	**TBR**	Top Back Right
24	**TBC**	Top Back Center
25	**TFLC**	Top Front Left of Center
26	**TFRC**	Top Front Right of Center
27	**TBLC**	Top Back Left of Center
28	**TBRC**	Top Back Right of Center
29	**BOC**	Bottom Center
30	**BFL**	Bottom Front Left
31	**BFR**	Bottom Front Right
32	**BFC**	Bottom Front Center
33	**BSL**	Bottom Side Left
34	**BSR**	Bottom Side Right
35	**BBL**	Bottom Back Left
36	**BBR**	Bottom Back Right
37	**BBC**	Bottom Back Center
38	**BFLC**	Bottom Front Left of Center
39	**BFRC**	Bottom Front Right of Center
40	**BBLC**	Bottom Back Left of Center
41	**BBRC**	Bottom Back Right of Center
42	**HPL**	Headphone Left

Ordinal	Acronym	Description
43	**HPR**	Headphone Right
44	**VFL**	Vibro-kinetic Front Left
45	**VFR**	Vibro-kinetic Front Right
46	**VBL**	Vibro-kinetic Back Left
47	**VBR**	Vibro-kinetic Back Right
48	**VFC**	Vibro-kinetic Front Center
49	**VBC**	Vibro-kinetic Back Center
50	**VSL**	Vibro-kinetic Side Left
51	**VSR**	Vibro-kinetic Side Right

The following figure places these spatial locations (except for the vibro-kinetic related locations) in a three-dimensional listening environment.

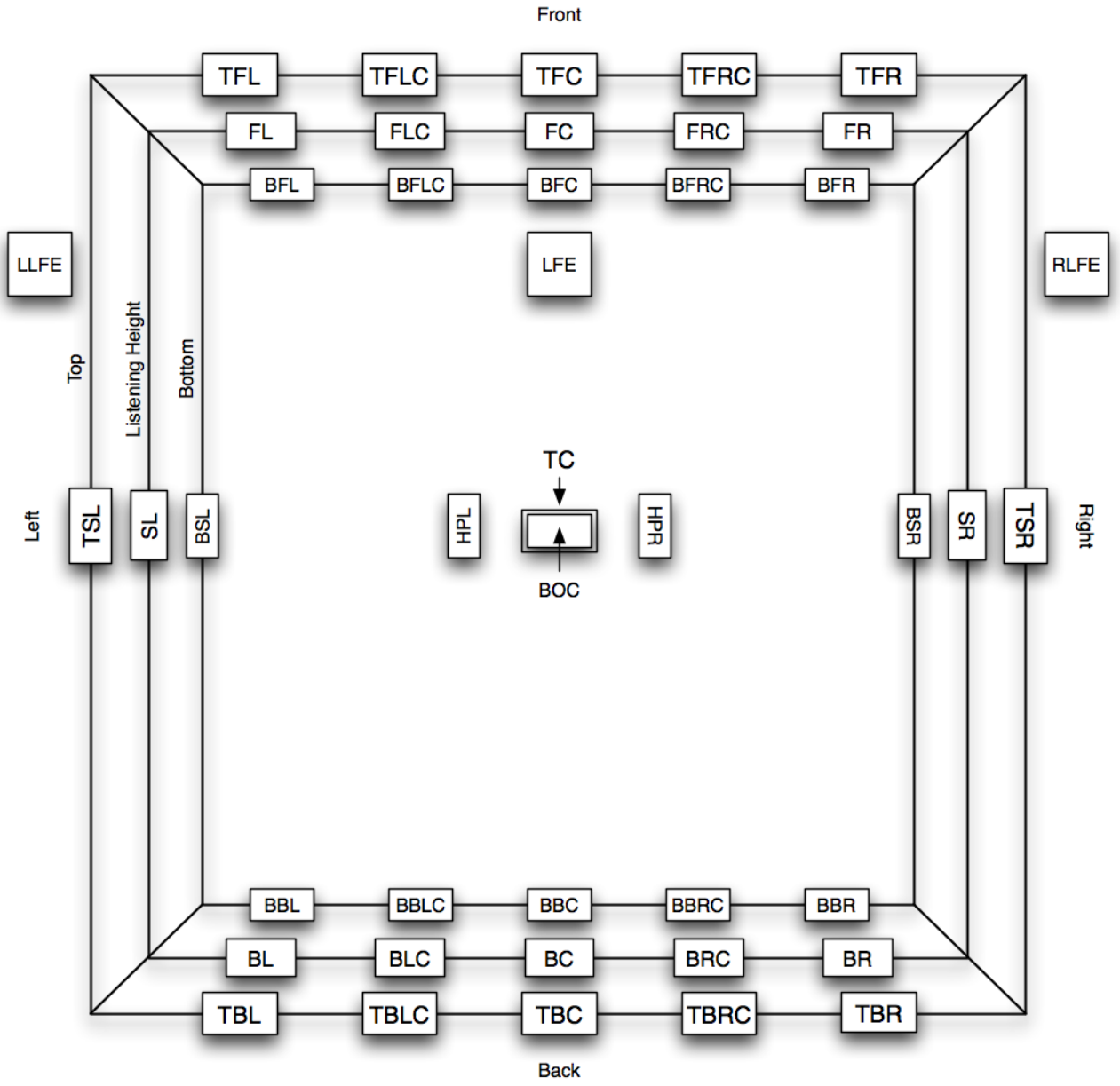


Figure 5-4: Three-dimensional Listening Environment

Besides the predefined AudioChannel Types, this specification allows for custom-defined AudioChannel Types. The AV Description Document shall then contain an ordered list of custom-defined type values.

The number of AudioChannels per AudioTrack as well as the spatial locations for those AudioChannels may vary for different AudioTracks in the same AudioCluster.

To be able to describe more complex cases in a manageable fashion, this specification imposes some limitations and restrictions on the ordering of AudioChannels in an AudioTrack and also in the AudioCluster.

- If there are AudioChannels present in an AudioTrack that correspond to some of the predefined AudioChannel Types (spatial positions) then they shall appear in the order specified in the ordered list of predefined AudioChannel Types. Refer to Table 5-2, “Predefined AudioChannel Types”. Any AudioChannels that correspond to custom-defined AudioChannel Types shall immediately follow the AudioChannels with predefined AudioChannel Types and they shall appear in the order as specified in the list of custom-defined AudioChannel Types in the AV Description Document (see Section 4.3, “AV Description Document”).

- This ordering automatically assigns a channel number to each AudioChannel in the AudioTrack: the first AudioChannel is channel 1, the second AudioChannel is channel 2, and so on up to the last AudioChannel in the AudioTrack.
- The default AudioTrack (main AudioTrack) shall come first in the AudioCluster and shall be AudioTrack one.
- Additional AudioTracks, if present, follow the default AudioTrack using a contiguously ascending numbering scheme.

5.4.2.1. Multi-channel Audio Example

Suppose an AVCluster contains 2 different AudioTracks for the same program, the first main AudioTrack containing 5.1 audio content, and the second AudioTrack containing 2.0 audio content. Then, the total number of AudioChannels in the AudioCluster will be 8 and, when entering the AVCore (through an Output Pin of an Input Terminal), the AudioChannels will be ordered as follows:

- The first six AudioChannels (numbered from one to six) constitute the first AudioTrack (AudioTrack 1) and contain the 5.1-channel audio content from the main AudioTrack (as intended by the content creator) where the first AudioChannel contains Front Left (FL), the second AudioChannel contains Front Right (FR), the third AudioChannel contains Front Center (FC), the fourth AudioChannel contains Side Left (SL), the fifth AudioChannel contains Side Right (SR), and the sixth AudioChannel contains the Low Frequency Effects (LFE).
- The next two AudioChannels (numbered one to two) constitute the second AudioTrack (AudioTrack 2) and contain the 2.0-channel audio content from the secondary AudioTrack (as intended by the content creator) where the first AudioChannel contains Front Left (FL), and the second AudioChannel contains Front Right (FR).

5.4.3. Multi-channel Metadata

Although not very common, there are cases where multi-channel metadata is desired. For example, it is possible that multiple metadata streams that contain information in the same language but are otherwise different need to be bundled with the associated video and audio content. As an example, consider the case where a media stream contains (besides video – possibly multi-channel – and audio – most likely multi-channel) both subtitles and director's commentaries in text form in multiple languages. All related MetadataChannels (subtitles and director's commentaries) in one language are grouped into one MetadataTrack.

To support multiple languages, several MetadataTracks may be grouped inside a single MetadataCluster.

Each MetadataChannel in a MetadataTrack has an associated MetadataChannel Type, which shall be unique for each MetadataChannel. The MetadataChannel Type fully identifies the MetadataChannel.

This specification defines an ordered list of 12 predefined MetadataChannel Types:

Table 5-3: Predefined MetadataChannel Types

Ordinal	Acronym	Description
1	**SUB**	Subtitles
2	**CAP**	Captions
3	**COM**	Commentaries
4	**TXT**	Text message
5	**KKE**	Karaoke
6	**RDS**	Radio Data System
7	**TTT**	Teletext
8	**EMM**	Emergency Message
9	**TCO**	Time Code
0	**GPS**	GPS Data
11	**NAV**	Navigation Data
12	**EPG**	EPG

Besides the predefined MetadataChannel Types, this specification allows for custom-defined MetadataChannel Types. The AV Description Document shall then contain an ordered list of custom-defined type values.

The number of MetadataChannels per MetadataTrack as well as the MetadataChannel Types for those MetadataChannels may vary for different MetadataTracks in the same MetadataCluster.

To be able to describe more complex cases in a manageable fashion, this specification imposes some limitations and restrictions on the ordering of MetadataChannels in a MetadataTrack and also in the MetadataCluster:

- If there are MetadataChannels present in a MetadataTrack that correspond to some of the predefined MetadataChannel Types, then they shall appear in the order specified in the ordered list of predefined MetadataChannel Types. Refer to Table 5-3, "Predefined MetadataChannel Types". Any MetadataChannels that correspond to custom-defined MetadataChannel Types shall immediately follow the MetadataChannels with predefined MetadataChannel Types and they shall appear in the order as specified in the list of custom-defined MetadataChannel Types in the AV Description Document (see Section 4.3, "AV Description Document").
- This ordering automatically assigns a channel number to each MetadataChannel in the MetadataTrack: the first MetadataChannel is channel 1, the second MetadataChannel is channel 2, and so on up to the last MetadataChannel in the MetadataTrack.
- The default MetadataTrack (main MetadataTrack) shall come first in the MetadataCluster and shall be MetadataTrack one.
- Additional MetadataTracks, if present, follow the default MetadataTrack using a contiguously ascending numbering scheme.

5.4.3.1. Multi-channel Metadata Example

Suppose an AVCluster contains 2 different MetadataTracks for the same program, the first main MetadataTrack containing Subtitles, Commentaries, and Teletext metadata, all in English, and the second MetadataTrack containing Subtitles metadata, in Dutch. Then, the total number of MetadataChannels in the MetadataCluster will be four and, when entering the AVCore (through an Output Pin of an Input Terminal), the MetadataChannels will be ordered as follows:

- The first three MetadataChannels (numbered one to three) constitute the first MetadataTrack (MetadataTrack 1) and contain the Subtitles (SUB), Commentaries (COM), and Teletext (TTT) content, all in English, from the main MetadataTrack (as intended by the content creator) in that order.
- The next MetadataChannel (numbered one) constitutes the second MetadataTrack (MetadataTrack 2) and contains the Subtitles (SUB) content from the secondary MetadataTrack (as intended by the content creator).

5.4.4. Raw Data

To support the case where the information flowing over a connection cannot be interpreted as a cluster of logical VideoChannels, AudioChannels or MetadataChannels, this specification defines an additional Cluster type called RawDataCluster. This type is mutually exclusive with all other Cluster types defined in this specification. It cannot co-exist in the same AVCluster with other Cluster types and its use is very restricted. It can only be used on connections between Entities that do not manipulate AV content. These Entities can only be Input Terminals and Output Terminals. The RawDataCluster can only be used to implement a pass-through for a data stream with unknown characteristics from an AVData In Entity to an AVData Out Entity with no processing whatsoever in between. All other use of the RawDataCluster is prohibited.

5.4.5. AVCluster Characteristics

An AVCluster is characterized by a number of properties:

VideoCluster:

- The number of VideoTracks in the VideoCluster. This number is implied by the number of `<videoTrack>` child elements of the `<videoCluster>` element in the VideoCluster Description.
- For each VideoTrack:
  - The number of VideoChannels in the VideoTrack. This number is implied by the number of child elements of the `<channels>` element in the VideoCluster Description.
  - A unique VideoChannel Type (observation location) for each VideoChannel in the VideoTrack: either predefined (OL001, OL002, etc.) or custom-defined.

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

AudioCluster:

- The number of AudioTracks in the AudioCluster. This number is implied by the number of `<audioTrack>` child elements of the `<audioCluster>` element in the AudioCluster Description.
- For each AudioTrack:
  - The number of AudioChannels in the AudioTrack. This number is implied by the number of child elements of the `<channels>` element in the AudioCluster Description.
  - A unique AudioChannel Type (spatial location) for each AudioChannel in the AudioTrack: either predefined (FL, FR, BL, LFE, etc.) or custom-defined.

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

  - Optionally, the language associated with the AudioTrack. The language shall be expressed in accordance with [RFC5646].

MetadataCluster:

- The number of MetadataTracks in the MetadataCluster. This number is implied by the number of `<metadataTrack>` child elements of the `<metadataCluster>` element in the VideoCluster Description.
- For each MetadataTrack:
  - The number of MetadataChannels in the MetadataTrack. This number is implied by the number of child elements of the `<channels>` element in the MetadataCluster Description.
  - A unique MetadataChannel Type for each MetadataChannel in the MetadataTrack: either predefined (SUB, CAP, etc.) or custom-defined.
  - Optionally, the language associated with the MetadataTrack. The language shall be expressed in accordance with [RFC5646].

RawDataCluster:

- Mutually exclusive with all other Cluster types.
- No descriptive information necessary.

There is a Cluster Description (CD) associated with each AVCluster that fully describes it.

5.5. Track Selectors and Channel Configurations

AVClusters travel over connections within the AVCore. The fact that an Output Pin is connected to an Input Pin indicates that the AVCluster can travel over that connection. An AVCluster may contain multiple VideoTracks, AudioTracks, and MetadataTracks simultaneously.

5.5.1. Track Selectors

In most cases, Units operate only on a single VideoTrack, AudioTrack, or MetadataTrack from each Input Pin at a time. Therefore, for each Input Pin of the Unit, there is a mechanism in place to select the VideoTrack, AudioTrack, and MetadataTrack from the AVCluster on that Input Pin on which the Unit will operate. The selected Tracks are called the Active Tracks.

If an Input Pin (on behalf of its Unit) needs to be able to indicate or select the Active VideoTrack from the incoming multi-track VideoCluster, it exposes an associated VideoTrack Selector Control. Likewise, if an Input Pin needs to be able to indicate or select the Active AudioTrack from the incoming multi-track AudioCluster, it exposes an associated AudioTrack Selector Control. Similarly, if an Input Pin needs to be able to indicate or select the Active MetadataTrack from the incoming multi-track MetadataCluster, it exposes an associated MetadataTrack Selector Control.

If an Input Pin does not expose a Track Selector, it 'inherits' the Track selection from the topologically closest Unit upstream from it that performs the Active Track selection (either implicit by construction or explicit by means of a Track Selector).

Track Selector Controls can be either Read-Only (the Unit decides by construction on which Track it operates) or Read-Write (Controller software can select the Active Tracks).

A VideoTrack Selector Control shall be constructed such that it supports the maximum number of VideoTracks that can ever be present in any potential VideoCluster that may be present on the associated Input Pin. Likewise, an AudioTrack Selector Control shall be constructed such that it supports the maximum number of AudioTracks that can ever be present in any potential AudioCluster that may be present on the associated Input Pin. Similarly, a MetadataTrack Selector Control shall be constructed such that it supports the maximum number of MetadataTracks that can ever be present in any potential MetadataCluster that may be present on the associated Input Pin.

A Track Selector is constructed such that it can select any of the available Tracks in the incoming AVCluster. The available Tracks are numbered from 1 up to the maximum number of Tracks supported by the Track Selector Control (p in Figure 5-5). The Track Selector Control selects the Active Track from the available Tracks (Track 2 in Figure 5-5) and offers this Active Track to the downstream part of the Unit for further processing.

For single-input Units, the processed Active Track is then presented at the Output Pin of the Unit together with all the unselected Tracks in the original order. In the example in Figure 5-5, Track 1 and Track 3 through p are simply passed through the Unit untouched, and the processed Active Track appears at the Output Pin as Track 2.

The following figure illustrates the concept of a Track Selector for single-input Units.

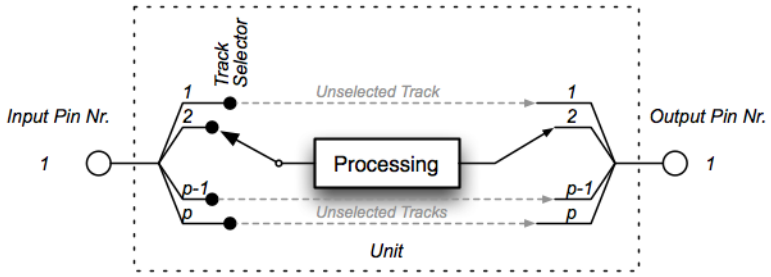


Figure 5-5: Track Selector for Single-input Units

For multi-input Units, each Input Pin may have Track Selectors as appropriate and follows the same rules as indicated for the Input Pin of a Single-Input Unit above, except for the following:

Multi-input Units combine the multiple incoming AVClusters into a single new outgoing AVCluster. The following situations can arise:

- A Selector Unit selects one of the available AVClusters at its Input Pins and presents the selected AVCluster unaltered at its Output Pin. A Selector Unit shall not have Track Selectors at its Input Pins and therefore, the Active Tracks are coming with the selected AVCluster and are thus determined by the Track selection mechanisms upstream from the selected Input Pin.
- A Router Unit selects from the available sets of VideoTracks, AudioTracks, and MetadataTracks at its Input Pins and constructs a new, potentially multi-track AVCluster at its Output Pin and discards all unused Tracks. In this case, the Input Pins of other Units connected to the Router Unit's Output Pin shall have the necessary Track Selectors to indicate or select the Active Tracks on which they want to operate.
- A Mixer Unit by definition creates at most a single VideoTrack and a single AudioTrack and therefore Track selection is implicit.

In general, the overall AVFunction topology is composed of several different signal paths that originate at the Input Terminals of the AVFunction, flow through the AVCore to eventually terminate in one of the Output Terminals of the AVFunction. One such signal path usually consists of several cascaded (potentially different) Units that offer certain AV functionality.

In most cases (but not all), the Active VideoTracks, AudioTracks, and MetadataTracks are selected at the beginning of the signal path and all cascaded Units in the signal path are intended to operate on the same Active Tracks. In this case, Track Selectors are not present on the Input Pins of the cascaded Units and they 'inherit' the Active Track(s) as selected by the Track Selector(s) of the first Unit in the cascade. The following figure provides an example where three (potentially different) single-input Units are cascaded.

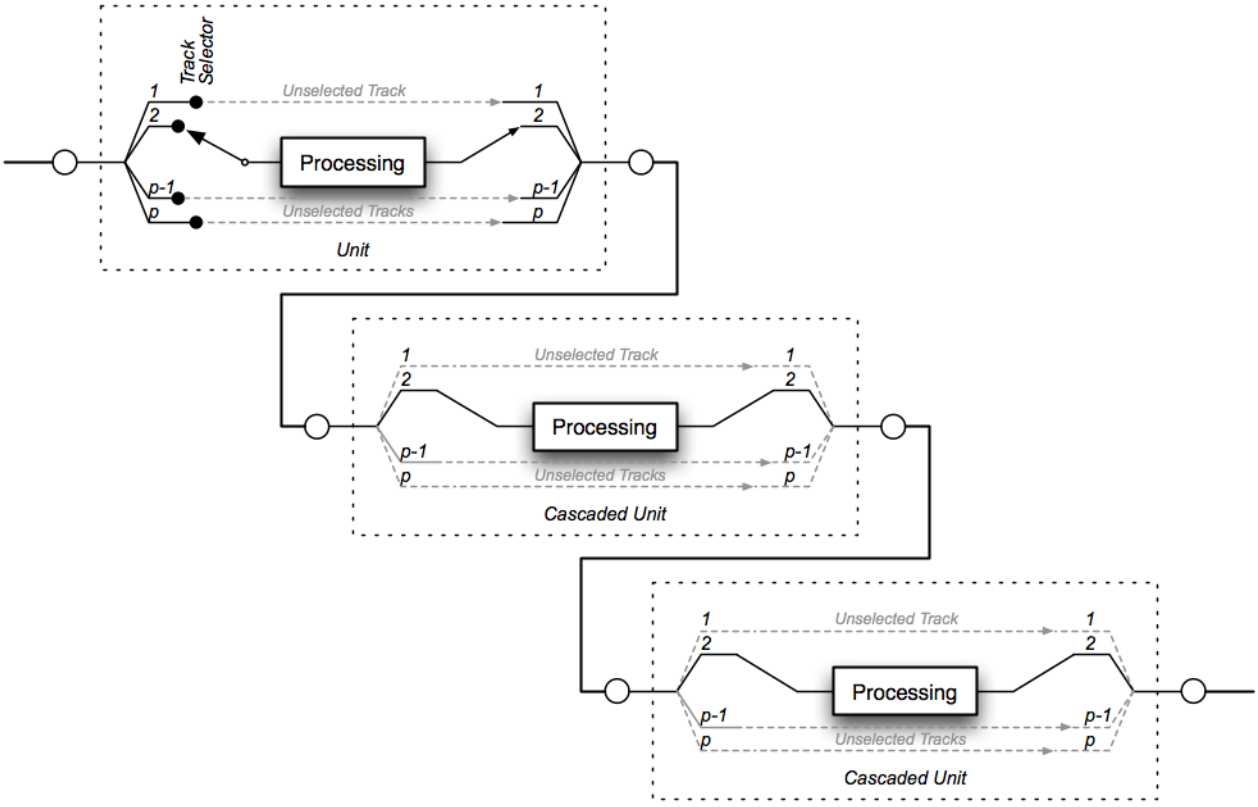


Figure 5-6: Cascaded Units

For single-input Units, if there is a need to manipulate AVControls on multiple Tracks at the same time, Units that contain the appropriate AVControls can be cascaded and each Unit can select a different Track and act upon it. The following figure illustrates this situation. Unit 1 and Unit 2 operate on Track 2 and Unit 3 operates on Track p-1 but still passes Track 2 as processed by Unit 1 and 2 into its outgoing AVCluster.

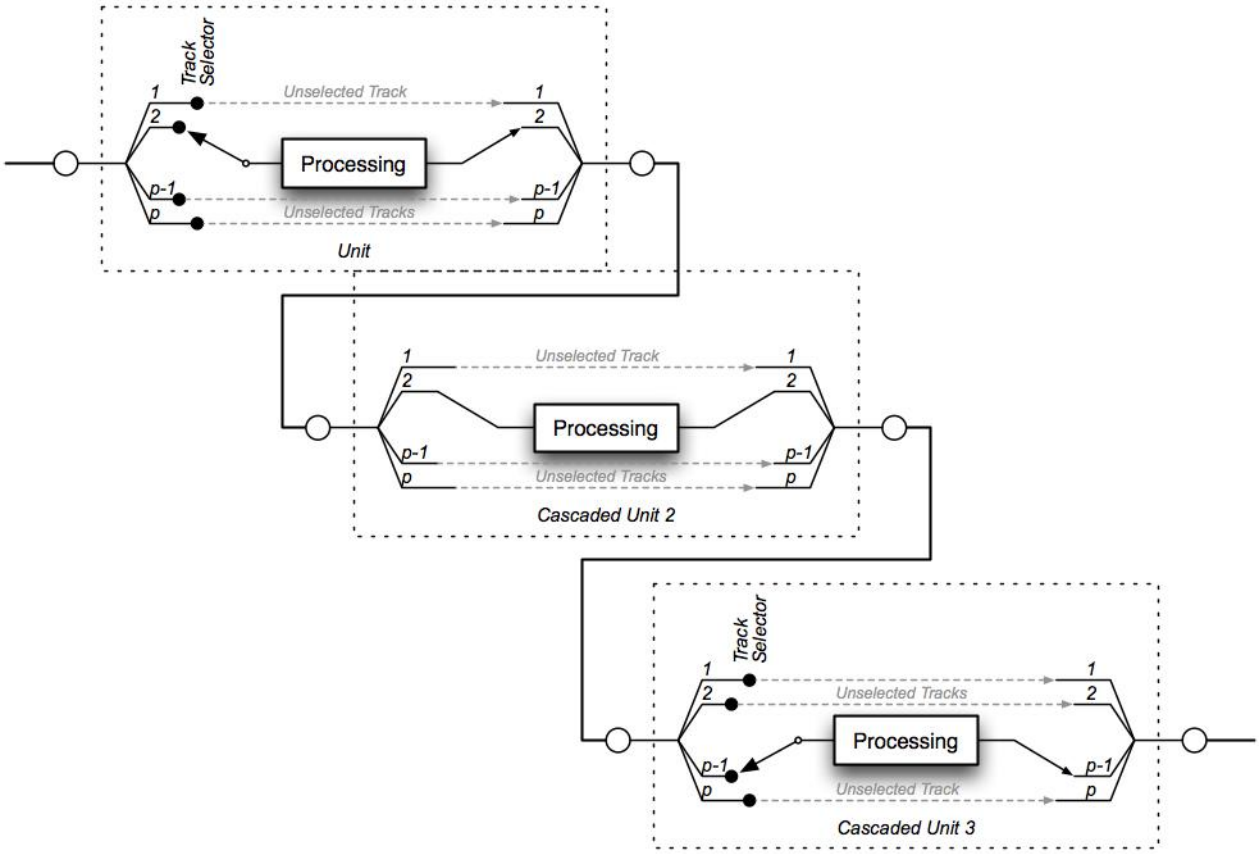


Figure 5-7: Cascaded Units Processing Different Tracks

The same chain of processing could be achieved by adopting the topology of Figure 5-8. However, this topology requires a Router Unit to recombine the separately processed tracks from Unit 1/2 and Unit 3. Note also how Router Unit Input Pin 1 inherits its Track selection from Unit 1 (through Unit 2) and Input Pin 2 inherits its Track selection from Unit 3.

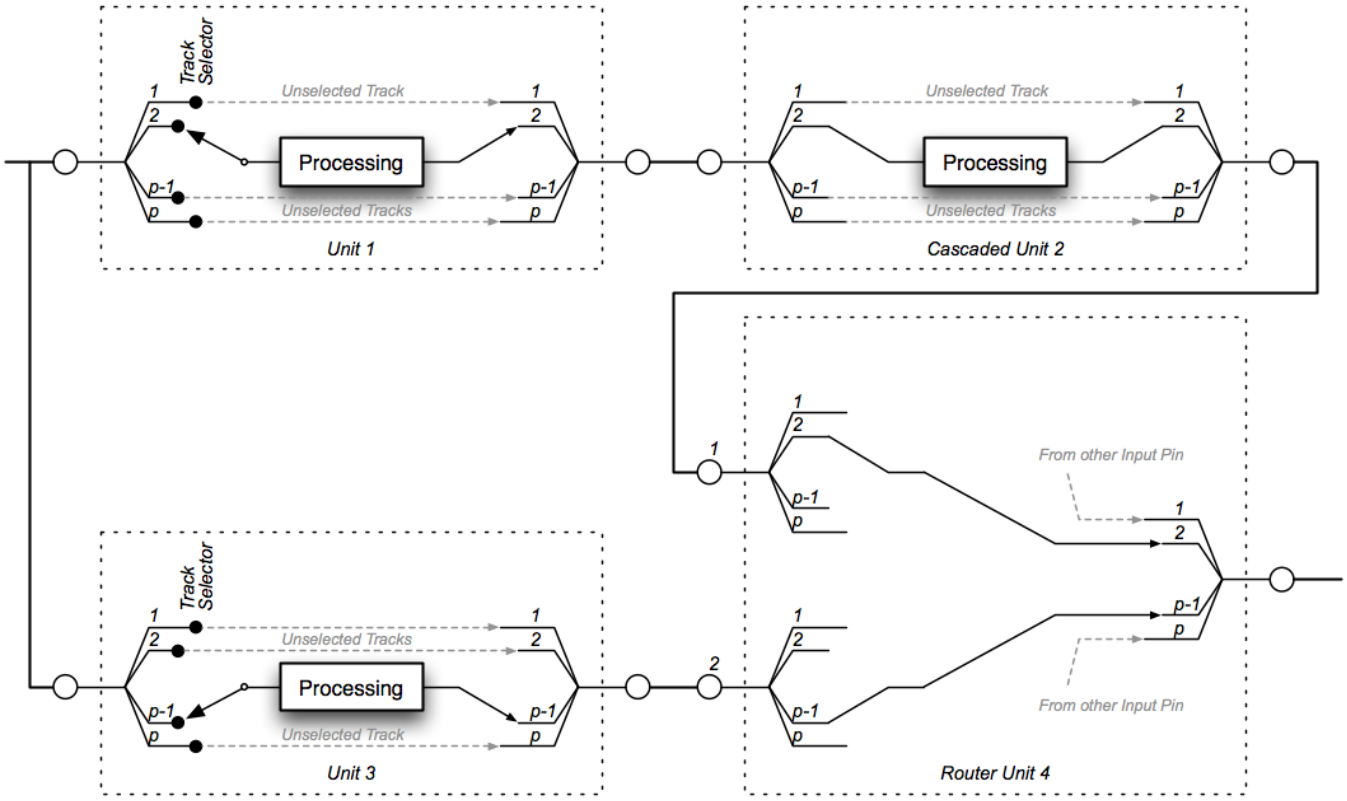


Figure 5-8: Parallel Units Processing Different Tracks

5.5.2. Channel Configurations

Once a Track is selected, AVControls located inside the Unit provide a means to influence or alter certain aspects of the VideoChannels, AudioChannels or MetadataChannels in the Active VideoTrack, AudioTrack, or MetadataTrack.

To describe the topology of the available AVControls inside a Unit, the concept of a Channel Configuration is introduced. There is an (explicit) Input Channel Configuration defined for each Input (Pin) of the Unit. There is an (explicit or implicit) Output Channel Configuration defined for each Output (Pin) of the Unit.

The following figure illustrates the Channel Configuration concept. For simplicity reasons, only Input Pin 2 is expanded to show an Input Channel Configuration. All other Input Pins have a similar Input Channel Configuration. Also for simplicity reasons, only one Channel Configuration is shown where, in reality, an Input Pin may have provisions for a VideoChannel, AudioChannel, and a MetadataChannel Configuration simultaneously and therefore also have a VideoTrack, AudioTrack, and MetadataTrack Selector present at the same time.

At each crossing of an input channel and an output channel, there can be one or more (different) AVControls present (indicated by the black dots in Figure 5-9).

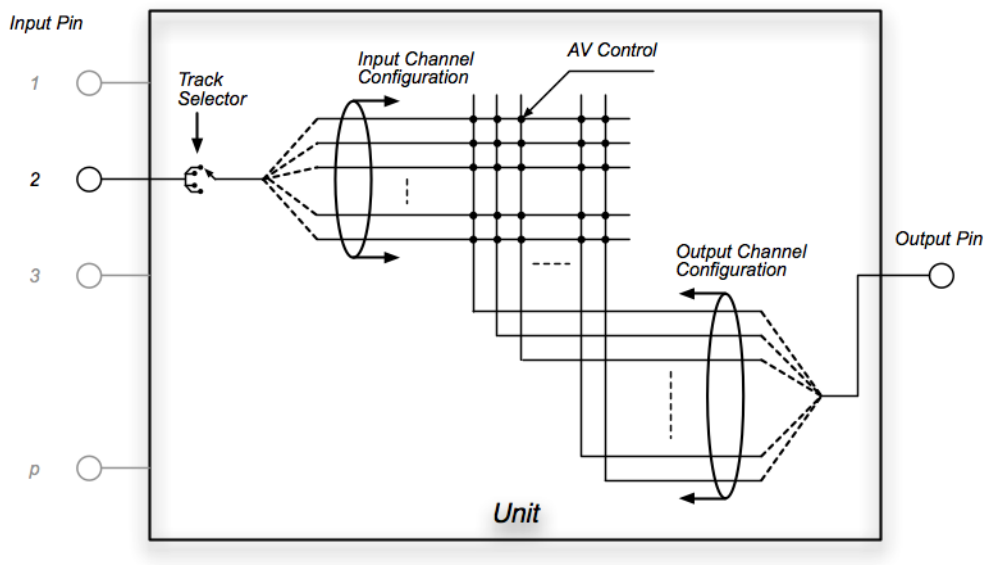


Figure 5-9: Channel Configurations

For single input Units that do not alter their Channel Configuration, the Input Channel Configuration is maintained throughout the Unit so that its Output Pin inherits the Input Channel Configuration for its Output Channel Configuration.

The maximum number of VideoChannels in a signal path can be up to 65,535. The maximum number of AudioChannels can also be up to 65,535 as can be the maximum number of MetadataChannels.

5.5.2.1. Input Channel Configuration

An Input Channel Configuration is characterized by a number of properties that are advertised through a corresponding Input Channel Configuration Description:

- Video input signal path:
  - The (optional) VideoTrack Selector Control Description.
  - The number of VideoChannels in the input video signal path. This number is implied by the number of child elements of the `<inputChannelConfig:videoPath:channels>` element in the Input Pin Description.
  - The VideoChannel Type (observation location) of each VideoChannel, either predefined (OL001, OL002, etc.) or custom-defined, organized in an ordered list:
    - If there are VideoChannels present in the video path that correspond to some of the predefined VideoChannel Types (observation locations), then they shall appear in the order specified in the ordered list of predefined VideoChannel Types. Refer to Table 5-1, “Predefined VideoChannel Types”. Any VideoChannels that correspond to custom-defined VideoChannel Types shall immediately follow the VideoChannels with predefined VideoChannel Types and they shall appear in the order as specified in the list of custom-defined VideoChannel Types in the AV Description Document.
    - The VideoChannels are then assigned a number starting from one up to the total number of VideoChannels in monotonically ascending order as they appear in the list. The VideoChannel number (instead of the VideoChannel Type) is used in Control Sequences to address a specific video AVControl inside a Unit. Channel number zero is used to address the Master Controls (if present).
- Audio input signal path:
  - The (optional) AudioTrack Selector Control Description.
  - The number of AudioChannels in the input audio signal path. This number is implied by the number of child elements of the `<inputChannelConfig:audioPath:channels>` element in the Input Pin Description.
  - The AudioChannel Type (spatial location) of each AudioChannel, either predefined (FL, FR, BL, LFE, etc.) or custom-defined, organized in an ordered list:
    - If there are AudioChannels present in the audio path that correspond to some of the predefined AudioChannel Types (spatial positions) then they shall appear in the order specified in the ordered list of

predefined AudioChannel Types. Refer to Table 5-2, “Predefined AudioChannel Types”. Any AudioChannels that correspond to custom-defined AudioChannel Types shall immediately follow the AudioChannels with predefined AudioChannel Types and they shall appear in the order as specified in the list of custom-defined AudioChannel Types in the AV Description Document.
    - The AudioChannels are then assigned a number starting from one up to the total number of AudioChannels in monotonically ascending order as they appear in the list. The AudioChannel number (instead of the AudioChannel Type) is used in Control Sequences to address a specific audio AVControl inside a Unit. Channel number zero is used to address the Master Controls (if present).
- Metadata input signal path:
    - The (optional) MetadataTrack Selector Control Description.
    - The number of MetadataChannels in the input metadata signal path. This number is implied by the number of child elements of the `<inputChannelConfig:metadataPath:channels>` element in the Input Pin Description.
    - The MetadataChannel Type of each MetadataChannel, either predefined (SUB, CAP, etc.) or custom-defined, organized in an ordered list:
        - If there are MetadataChannels present in a metadata path that correspond to some of the predefined metadata types, then they shall appear in the order specified in the ordered list of predefined metadata types. Refer to Table 5-3, “Predefined MetadataChannel Types”. Any MetadataChannels that correspond to custom-defined MetadataChannel Types shall immediately follow the MetadataChannels with predefined MetadataChannel Types and they shall appear in the order as specified in the list of custom-defined MetadataChannel Types in the AV Description Document.
        - The MetadataChannels are then assigned a number starting from one up to the total number of MetadataChannels in monotonically ascending order as they appear in the list. The MetadataChannel number (instead of the MetadataChannel Type) is used in Control Sequences to address a specific metadata AVControl inside a Unit. Channel number zero is used to address the Master Controls (if present).

5.5.2.2. Output Channel Configuration

An Output Channel Configuration Description is identical to an Input Channel Configuration Description except that it does not contain any Track Selector Control Description information (Output Pins do not have associated Track Selector Controls):

- Video output signal path:
    - The number of VideoChannels in the output video signal path. This number is implied by the number of child elements of the `<outputChannelConfig:videoPath:channels>` element in the Output Pin Description.
    - The VideoChannel Type (observation location) of each VideoChannel, either predefined (OL001, OL002, etc.) or custom-defined, organized in an ordered list (see above).
    - The native aspect ratio of the output video signal path
- Audio output signal path:
    - The number of AudioChannels in the output audio signal path. This number is implied by the number of child elements of the `<outputChannelConfig:audioPath:channels>` element in the Output Pin Description.
    - The AudioChannel Type (spatial location) of each AudioChannel, either predefined (FL, FR, BL, LFE, etc.) or custom-defined, organized in an ordered list (see above).
- Metadata output signal path:
    - The number of MetadataChannels in the output metadata signal path. This number is implied by the number of child elements of the `<outputChannelConfig:metadataPath:channels>` element in the Output Pin Description.
    - The MetadataChannel Type of each MetadataChannel, either predefined (SUB, CAP, etc.) or custom-defined, organized in an ordered list (see above).

5.6. AVControl Hierarchical Accessing Scheme

To access a particular AVControl inside an Entity, the following information is needed:

- The EntityID of the Entity that contains the targeted AVControl (EID)
- The Property of the targeted AVControl that is to be manipulated (PROP)
- The Control Selector value to uniquely identify the type of AVControl that is targeted (CS)
- IPN-ICN-OCN Triplet: Input Pin Number-Input Channel Number-Output Channel Number.

For details about EntityID, Property, Control Selector and the IPN-ICN-OCN triplet and their allowed values, see Section 9.3.1.1, "Message Fields".

The following figure illustrates the hierarchical addressing scheme of AVControls described above. For simplicity, only one type of signal path (video, audio, or metadata) is depicted:

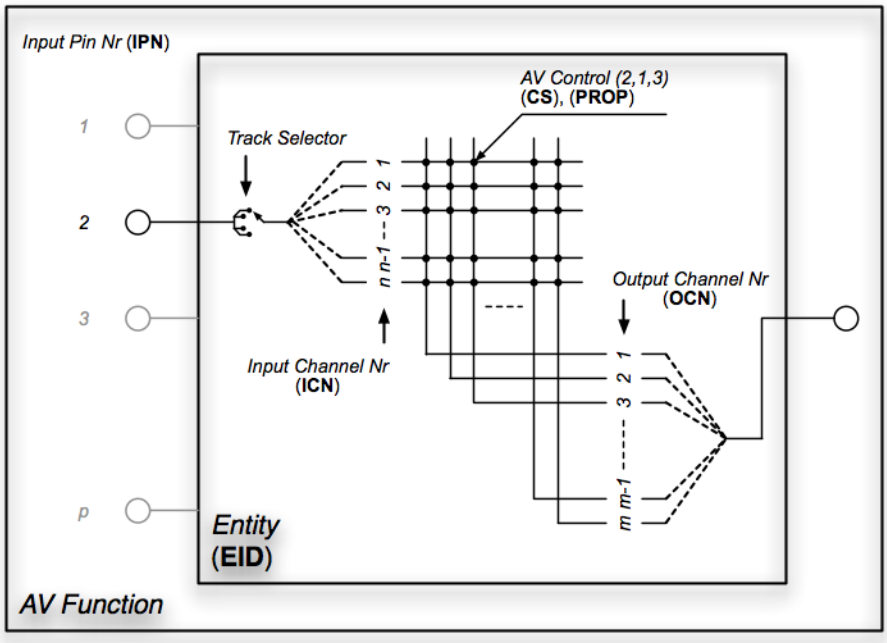


Figure 5-10: Hierarchical Accessing Scheme for AVControls

5.7. Relationship between AVClusters and Channel Configurations

As described above, each flow of AVStream within the AVCore is modeled via the AVCluster concept and is characterized by its AVCluster Description, indicating the number of VideoChannels, AudioChannels, VideoTracks, AudioTracks, etc. that make up the entire AVCluster. The AVCluster configuration may change dynamically over time, depending on changes in settings outside the AVCore. For example, the Alternate Setting on an AVData Streaming-In interface may change causing a change in the AVCluster that enters the AVCore through the associated Input Terminal's Output Pin.

Inside the AVCore, the AVClusters flow over a set of connections and through a set of Units, starting from the Output Pin of an Input Terminal and finally arriving at the Input Pin of an Output Terminal, undergoing all kinds of transformations and processing steps in between.

Although AVClusters, connections, Terminals and Units are logical concepts, there is still an underlying physical hardware/firmware/software implementation that eventually needs to perform the tasks as indicated by the logical building blocks of the AVCore. This underlying system is built by design to support a certain set or range of AVCluster configurations (those that are supported by the Alternate Settings of an AVData Entity). The Channel Configurations on the different Input and Output Pins of the logical building blocks therefore are static in nature since they logically represent the underlying device implementation.

Channel Configuration Descriptions provide information about the configuration of the signal paths inside the Units and Terminals (and GraphicsEngine Entity). AVCluster Descriptions provide information about the configuration of the AVClusters that are currently flowing over those signal paths.

In general, a Channel Configuration is designed to be a superset (the union) of all the AVCluster **Track** configurations that can flow over the signal path. Indeed, an Input Pin first selects the Active Track by means of the Track Selector and the Active Track is subsequently further processed by the Unit. For example, if an audio signal path is designed to handle 5.1-channel audio (FL, FR, FC, SL, SR, LFE), then any AudioTrack within an AudioCluster that contains a subset of these 6 channels can successfully flow over the 5.1-channel audio path.

However, in some cases, there can be exceptions to this general principle. Part of the underlying system may be designed to only support a subset of the AVCluster configurations that are supported. For example, an AVData Streaming Interface may be designed to accept 3D2 video content but the AVCore may only be able to handle 2D content.

The following Section describes how the mapping between the Active Track and the signal path's Channel Configuration takes place in all cases.

5.7.1. Mapping between AVCluster Tracks and Channel Configurations

AVData In Entities can have multiple Alternate Settings, each representing a different 'mode of operation' of the Entity. For instance, an AVData Streaming-In interface representing a USB OUT pipe could have one Alternate Setting that is used when the AV content is MPEG-2 encoded and another Alternate Setting that is used when the AV content is MPEG-4 encoded. In the MPEG-2 case, the stream may have a single VideoChannel, 5.1 AudioChannels, and zero MetadataChannels, whereas in the MPEG-4 case, there may be multiple VideoTracks and VideoChannels, multiple AudioTracks and AudioChannels, and possibly multiple MetadataTracks and MetadataChannels. Switching from one Alternate Setting to another can therefore effectively change the number of actual VideoChannels AudioChannels and MetadataChannels in the interface and possibly also the configurations of those Channels. Consequently, the configuration of the AVCluster entering the AVCore through the Output Pin of the Input Terminal that represents the AVData Streaming-In interface will then also change. Likewise, an AVData Out Entity may have multiple Alternate Settings, each representing a different 'mode of operation' of the interface. For example, an AVData Audio Streaming-Out interface representing a USB IN pipe could have one Alternate Setting that the Controller uses when it wants to retrieve a stereo version of the audio content and another Alternate Setting that the Controller uses when it wants to retrieve a full 5.1 version of the same audio content.

In general, any Output Pin inside the AVCore can potentially carry different AVCluster Track configurations at a given point in time. Therefore, it must be possible to map each of these AVCluster Track configurations onto the signal path inside the Entity that is connected to that Output Pin. If there is no explicit Up-Down-mix Processing Unit present, then the Mapping Rules described in the following section are used to map all of the possible AVCluster Track configurations onto a single signal path inside the connected Entity.

5.7.2. Mapping Rules

The following Mapping rules apply equally to VideoChannel, AudioChannel, and MetadataChannel Types.

The Mapping Rules consist of the following (see Figure 5-11):

- If a particular Channel Type is available in the Active Track of the AVCluster (ch x) and is also available in the signal path's Channel Configuration (CH X), ch x shall be mapped onto CH X.
- If a particular Channel Type is not available in the Active Track of the AVCluster (ch y) but is available in the signal path's Channel Configuration (CH Y), CH Y carries implementation dependent information. However, it is recommended that for Video, this means an equally colored still image, preferably black (RGB=0x000000); for audio, this means silence; and for Metadata this means no information.
- If a particular Channel Type is available in the Active Track of the AVCluster (ch z) but is not available in the signal path's Channel Configuration (CH Z), ch z shall be discarded.

These rules are applicable to both predefined Channel Types and custom-defined Channel Types.

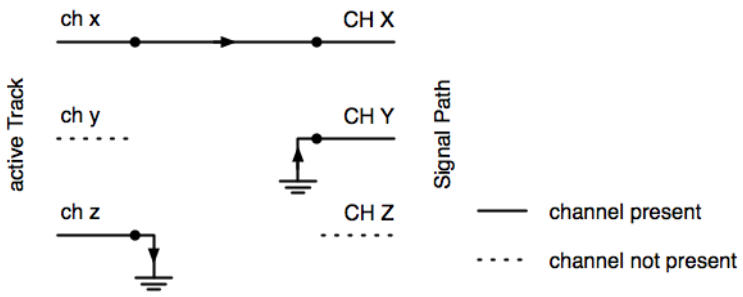


Figure 5-11: Mapping Rules

In addition, for the VideoCluster, aspect ratio shall always be preserved when mapping the Active VideoTrack onto the video signal path by adding uniformly colored borders either on the left and right side or at the top and bottom of the image if there is a mismatch between the aspect ratio of the Active VideoTrack and the aspect ratio of the video signal path.

5.7.3. Mapping between Differing Channel Configurations

The previous section addressed the issues related to mapping VideoTracks, AudioTracks, and MetadataTracks onto the underlying video, audio, and metadata signal paths. A similar issue exists when connecting an Output Pin of an Entity to an Input Pin of another Entity and the Output Channel Configuration of the Output Pin does not exactly match the Input Channel Configuration of the Input Pin. The same mapping rules as established in Section 5.7.2, "Mapping Rules" apply but now between Output and Input Channel Configuration as follows:

- If a particular Channel Type is available in the Output Channel Configuration (ch x) and is also available in the Input Channel Configuration (CH X), ch x shall be mapped onto CH X.
- If a particular Channel Type is not available in the Output Channel Configuration (ch y) but is available in the Input Channel Configuration (CH Y), CH Y carries implementation dependent information. However, it is recommended that for video, this means an equally colored still image, preferably black (RGB=0x000000); for audio, this means silence; and for Metadata this means no information.
- If a particular Channel Type is available in the Output Channel Configuration (ch z) but is not available in the Input Channel Configuration (CH Z), ch z shall be discarded.

These rules are applicable to both predefined Channel Types and custom-defined Channel Types.

In addition, for the video signal path, aspect ratio shall always be preserved when mapping the output video signal path onto the input video signal path by adding uniformly colored borders either on the left and right side or at the top and bottom of the image if there is a mismatch between the aspect ratio of the output video path and the aspect ratio of the input video signal path.

As an example, consider a Feature Unit that can process 2D video content and 5.1 audio content and its Output Pin is connected to – among others – another Feature Unit that can only process 2 AudioChannels to perform some specific processing on the SL and SR AudioChannels, for example. In this case, the second Feature Unit simply discards the video content and only picks up the Side Left and Side Right AudioChannels, discarding all other AudioChannels.

Note: If Feature Unit 2 only supports 2 Channels but still wants to use all of the available audio content, an explicit Converter Unit shall be inserted between Feature Unit 1 and 2 that converts the 5.1 audio content into 2.0 'enriched' content using some implementation-specific algorithm. See Section 6.9, "Converter Unit" for more details on Converter Units.

5.8. Still Images

Within the AVCore, where complete abstraction is made of the physical characteristics of the AVStreams, still images are treated very similar to true streaming video. In general, a still image is conceptually considered to be a (potentially multi-channel, multi-viewpoint) VideoStream that consists of a (possibly lengthy) repetition of the same frame information (the still image stream). Bundled in an AVCluster, still image streams may be accompanied by an AudioCluster and/or MetadataCluster as appropriate. They enter the AV Core through Input Terminals and leave the AV Core through Output Terminals. These Terminals represent AVData Entities that have still image capabilities.

6. Entity Definitions

6.1. Input Terminal

The Input Terminal (IT) is used to interface between the AVCore's outside world and Units and Terminals in the AVCore. It serves as a receptacle for AV information flowing into the AVCore. Its function is to represent a source of incoming data after this data has been properly extracted from the original AVStream into the separate Tracks and Channels that are embedded in this AVStream (the decoding process). The AV content is then organized into an AVCluster (as a logical concept) and leaves the Input Terminal through a single Output Pin.

An Input Terminal represents an input to the AVCore originating from an AVData In Entity. The Input Terminal is the logical representation within the AVCore of the physical AVData In Entity and there is always a one-to-one relationship between the Input Terminal and its associated AVData Entity.

In between the AVData In Entity and the Input Terminal there is always a decoder present (but accessed as part of the AVData In Entity), even if the decoding process is trivial and does not need Controller interaction (see Section 6.13.1, "Decoders" for more details).

The Controller needs to use both the AVData In Entity Description and the Input Terminal Description to get a full understanding of the characteristics and capabilities of the Input Terminal and what it represents.

The Input Terminal provides AVControls for the following features:

- Cluster (Section 9.9.1.1)

The Input Terminal shall maintain audio/video/metadata synchronization at its output at all times.

The symbol for the Input Terminal can be found in the following figure:

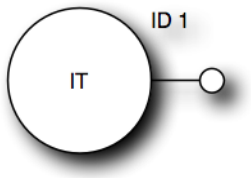


Figure 6-1: Input Terminal Icon

6.2. Output Terminal

The Output Terminal (OT) is used to interface between Units and Terminals inside the AVCore and the outside world. It serves as an outlet for AV information, flowing out of the AVCore. Its function is to represent a sink of outgoing data before this data is properly packed from the original separate logical Channels and Tracks into the outgoing AVStream (the encoding process). The AVCluster enters the Output Terminal through a single Input Pin. There is no VideoTrack, AudioTrack, or MetadataTrack Selector Control associated with the Input Pin of an Output Terminal. It is assumed that the information carried in the AVCluster entering the Output Terminal shall be sent in its entirety to the Entity that the Output Terminal represents. A Router Unit may be used to adjust the composition of the AVCluster just before it enters the Input Pin of the Output Terminal.

An Output Terminal represents an output from the AVCore to an AVData Out Entity. The Output Terminal is the logical representation within the AVCore of the physical AVData Out Entity and there is always a one-to-one relationship between the Output Terminal and its associated AVData Entity.

In between the Output Terminal and the AVData Out Entity there is always an encoder present (but accessed as part of the AVData Out Entity), even if the encoding process is trivial and does not need Controller interaction (see Section 6.13.2, Encoders" for more details).

The Controller needs to use both the AVData Out Entity Description and the Output Terminal Description to get a full understanding of the characteristics and capabilities of the Output Terminal and what it represents.

The Output Terminal provides no AVControls.

The Output Terminal shall maintain audio/video/metadata synchronization when delivering content to its associated AVData Entity at all times.

The symbol for the Output Terminal can be found in the following figure:

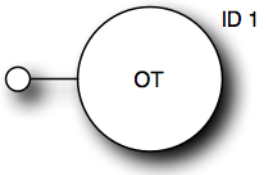


Figure 6-2: Output Terminal Icon

6.3. Mixer Unit

The Mixer Unit (MU) transforms a number of logical Input Channels into a number of logical Output Channels. The Input Channels are grouped into one or more AVClusters. Each cluster enters the Mixer Unit through a separate Input Pin.

Note: Not all AVClusters shall necessarily contain a VideoCluster, AudioCluster, and MetadataCluster. They are all optional components of an AVCluster.

The maximum number of Input Pins for a Mixer Unit is 65,535. If an Input Pin can accept multi-track video content, then an associated VideoTrack Selector Control may be present to indicate or select the Active VideoTrack. Likewise, if an Input Pin can accept multi-track audio content, then an associated AudioTrack Selector Control may be present to indicate or select the Active AudioTrack. MetadataTracks cannot be selected or processed by the Mixer Unit.

A Mixer Unit differs fundamentally from other Units in that, besides the Active VideoTrack and Active AudioTrack, all MetadataTracks, and any additional VideoTracks and AudioTracks (besides the Active VideoTrack and AudioTrack) present in any of the incoming AVClusters are discarded by the Mixer Unit and will not be found in the outgoing AVCluster. The Mixer Unit acts as an endpoint for all of the incoming AV information. It creates 'new' content from the incoming AVClusters by composing and rendering of the incoming Active VideoTracks onto an n-dimensional canvas and by mixing the incoming Active AudioTracks into a new outgoing AudioTrack. The outgoing AVCluster can therefore only have one VideoTrack and one AudioTrack and leaves the Mixer Unit through a single Output Pin. The outgoing AVCluster will never contain a MetadataTrack.

The Mixer Unit shall maintain audio/video synchronization at its output at all times.

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

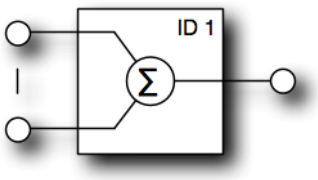


Figure 6-3: Mixer Unit Icon

6.3.1. Video Mixer

This version of the specification does not provide support for this item.

6.3.2. Audio Mixer

Any AudioChannel in any of the incoming AudioClusters can potentially be mixed into any AudioChannel of the outgoing AudioCluster.

The Audio Mixer provides AVControls for the following features:

- AudioTrack Selector (Section 9.10.2.1)
- Level (Section 9.10.2.2)

If n_i is the number of AudioChannels in an AudioTrack of an incoming AudioCluster i, and p is the number of Input Pins of the Mixer Unit, then n, the total number of Active AudioChannels in the Mixer Unit, is calculated as:

$$n = \sum_{i=1}^{p} n_i$$

and if m is the number of AudioChannels in the outgoing AudioTrack, then there is a two-dimensional array ($n \cdot m$) of Level Controls in the Mixer Unit. (n_i may be zero for certain values of i, i.e. an AudioCluster may be missing from some of the incoming AVClusters.)

A particular Level Control is identified using the triplet:

- Input Pin Number (IPN)
- Channel Number in the input AudioTrack (ICN)
- Channel Number in the output AudioTrack (OCN)

Figure 6-4 illustrates the concept:

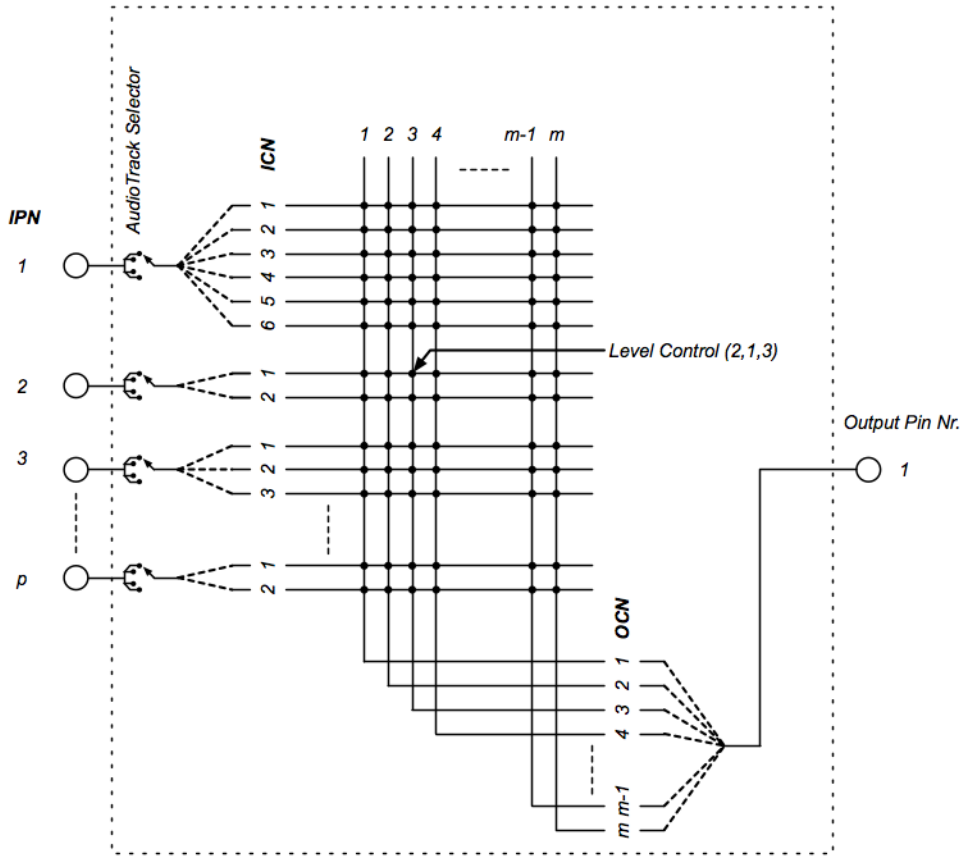


Figure 6-4: Audio Mixer Level Control Array

Not all of the Level Controls have to be physically implemented. A Level Control can be always on (the Input Channel is fully mixed into the Output Channel) or always off (the Input Channel is not mixed into the Output Channel at all). Level Controls can also be non-programmable (read-only).

6.4. Metadata Unit

The Metadata Unit (DU) provides mixing capabilities of the MetadataChannels in the Active MetadataTrack onto the video content of the Active VideoTrack (if present) of the incoming AVCluster. The AVCluster enters the Metadata Unit through a single Input Pin and leaves the Unit through a single Output Pin. If the Input Pin can accept multi-track video content, then an associated VideoTrack Selector Control may be present to indicate or select the Active VideoTrack. The Input Pin may have an associated MetadataTrack Selector Control to indicate or select the Active MetadataTrack. AudioTracks cannot be selected or processed by the Metadata Unit.

The Metadata Unit shall maintain audio/video/metadata synchronization at its output at all times.

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

Figure 6-5: Metadata Unit Icon

This version of the specification does not provide support for this item.

6.5. Selector Unit

The Selector Unit (SU) selects from n incoming AVClusters and routes one of them to the single output AVCluster unaltered. It represents a multi-channel source selector, capable of selecting between n sources. It has n Input Pins and a single Output Pin. The maximum number of Input Pins for a Selector Unit is 65,535. There is no Input Channel Configuration associated with the Input Pins of the Selector Unit. Likewise, there is no Output Chanel Configuration associated with the Output Pin. Since the selected AVCluster is simply passed from an Input Pin to the Output Pin, there is no VideoTrack, AudioTrack, or MetadataTrack Selector Control associated with the Input Pins of a Selector Unit.

The Selector Unit provides support for the following AVControl:

- Selector (Section 9.12.1)

The Selector Unit shall maintain audio/video/metadata synchronization at its output at all times.

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

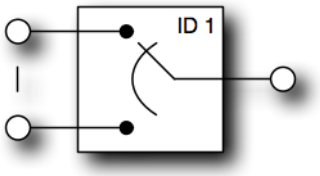


Figure 6-6: Selector Unit Icon

6.6. Feature Unit

The Feature Unit (FU) provides basic manipulation of multiple *single-parameter* VideoControls and AudioControls on the VideoChannels of the Active VideoTrack and the AudioChannels of the Active AudioTrack of the incoming AVCluster. The AVCluster enters the Feature Unit through a single Input Pin and leaves the Unit through a single Output Pin. If the Input Pin can accept multi-track video content, then an associated VideoTrack Selector Control may be present to indicate or select the Active VideoTrack. Likewise, if the Input Pin can accept multi-track audio content, then an associated AudioTrack Selector Control may be present to indicate or select the Active AudioTrack. Unselected VideoTracks and AudioTracks are passed unaltered from the incoming AVCluster to the outgoing

AVCluster. MetadataTracks cannot be selected or processed by the Feature Unit. They too are passed unaltered from the incoming AVCluster to the outgoing AVCluster.

The Active VideoTrack and AudioTrack are mapped onto the Feature Unit's Input Channel Configuration as described in Section 5.7.2, "Mapping Rules".

Support for any VideoControl or AudioControl listed below is optional. The Feature Unit Description reports what AVControls are present for every VideoChannel and AudioChannel in the Feature Unit (including the Master Channel).

AVControls that support an automatic mode shall have an associated AVControl with an on/off state. If the auto setting for a particular AVControl is supported and set to the on state, the Feature Unit shall provide automatic adjustment of the AVControl, and subsequent GET Sequences to the related AVControl shall reflect the automatically set value. Attempts to programmatically adjust the related AVControl shall result in an Error Response Message in auto mode. When leaving the auto mode, the related AVControl shall remain at the value that was in effect just before the transition.

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.

The following figure illustrates the internal configuration of a Feature Unit with respect to its AVControls.

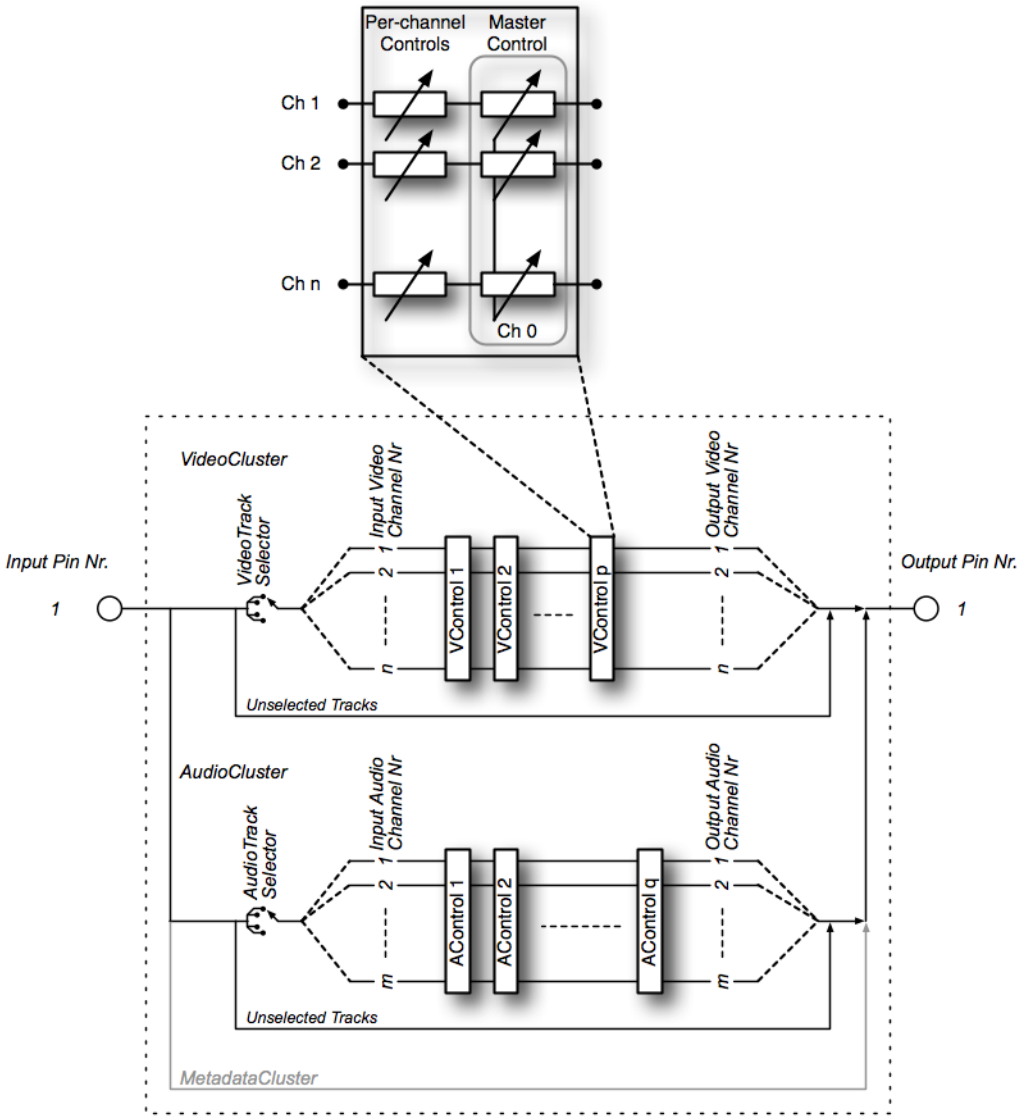


Figure 6-7: Feature Unit Internal Configuration

The Feature Unit shall maintain audio/video/metadata synchronization at its output at all times.

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

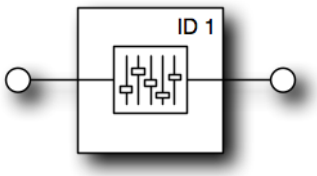


Figure 6-8: Feature Unit Icon

6.6.1. Feature Unit VideoControls

The Feature Unit provides VideoControls for the following features:

- VideoTrack Selector (Section 9.13.1.1)
- Brightness (Section 9.13.1.2)

- Contrast (Section 9.13.1.3)

  Note: Although this specification allows for per-channel Brightness and Contrast Controls, it is recommended that implementations only use Master Brightness and Contrast Controls so that corrections are applied to all Channels simultaneously.

6.6.2. Feature Unit AudioControls

For each AudioChannel, the Feature Unit provides AudioControls for the following features:

- AudioTrack Selector (section 9.13.2.1)
- Mute (Section 9.13.2.2)
- Volume (Section 9.13.2.3)
- Bass (Section 9.13.2.4)
- Mid (Section 9.13.2.5)
- Treble (Section 9.13.2.6)
- Graphic Equalizer (Section 9.13.2.7)
- Delay (Section 9.13.2.8)
- Bass Boost (Section 9.13.2.9)
- Loudness (Section 9.13.2.10)
- Input Gain (Section 9.13.2.11)
- Automatic Input Gain (Section 9.13.2.12)
- Input Gain Pad (Section 9.13.2.13)
- Phase Inverter (Section 9.13.2.14)

6.7. Effect Unit

An Effect Unit (EU) is either video-only or audio-only.

6.7.1. Video Effect Unit

The Video Effect Unit (VEU) provides advanced manipulation on a per-channel basis of a multi-parameter VideoControl on the VideoChannels of the Active VideoTrack of the incoming AVCluster. The AVCluster enters the Video Effect Unit through a single Input Pin and leaves the Unit through a single Output Pin. The Input Pin may have an associated VideoTrack Selector Control to indicate or select the Active VideoTrack.

The Active VideoTrack is mapped onto the Video Effect Unit's Input Channel Configuration as described in Section 5.7.2, "Mapping Rules".

Unselected VideoTracks are passed unaltered from the incoming AVCluster to the outgoing AVCluster. Likewise, an AudioCluster or MetadataCluster present in the input AVCluster is passed unaltered to the output AVCluster.

The following figure illustrates the internal configuration of a Video Effect Unit with respect to its AVControls.

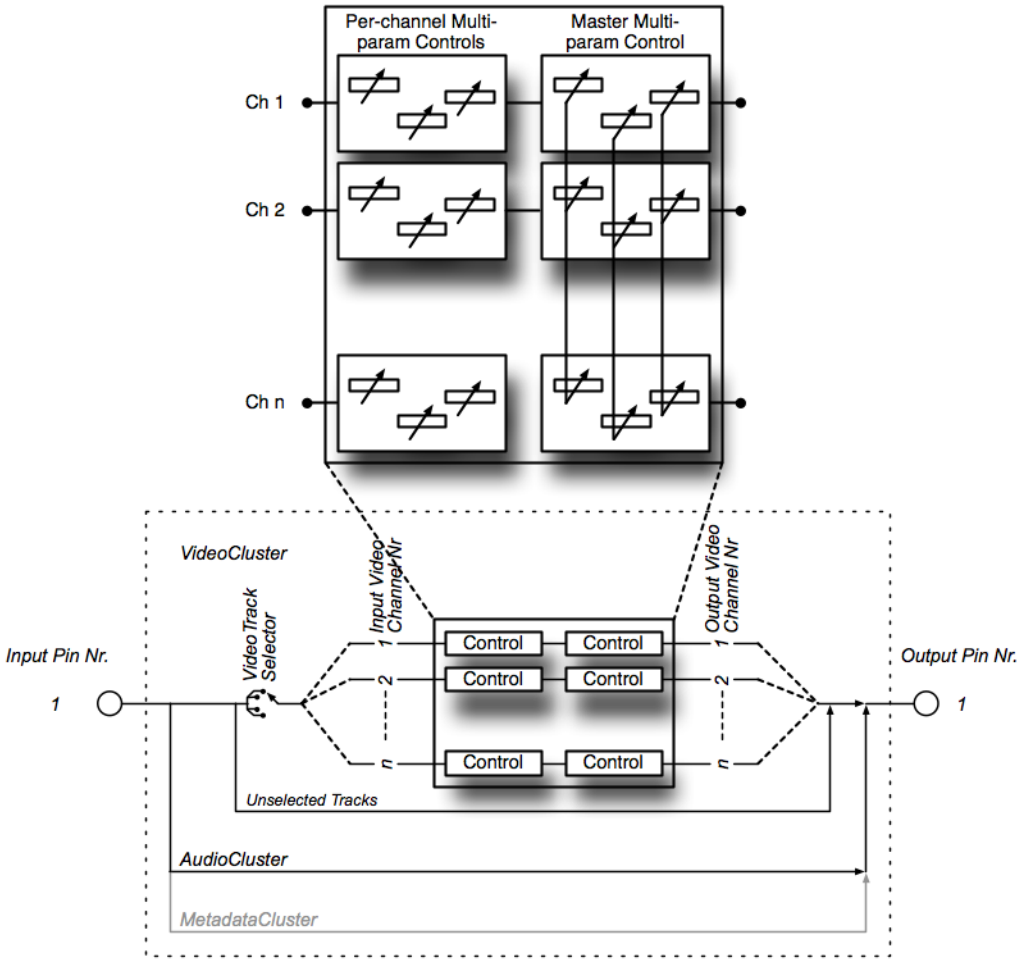


Figure 6-9: Video Effect Unit Internal Configuration

The Video Effect Unit shall maintain audio/video/metadata synchronization at its output at all times.

At this time, this specification only supports the Scaling Video Effect Unit. Later extensions may incorporate Effect Units such as Color Correction, Level Adjustment, Enhancement, etc.

6.7.1.1. Scaling Effect Unit

This version of the specification does not provide support for this item.

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

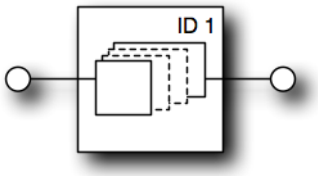


Figure 6-10: Scaling Effect Unit Icon

6.7.2. Audio Effect Unit

The Audio Effect Unit (AEU) provides advanced manipulation on a per-channel basis of a multi-parameter AudioControl on the AudioChannels of the Active AudioTrack of the incoming AVCluster. The AVCluster enters the

Audio Effect Unit through a single Input Pin and leaves the Unit through a single Output Pin. The Input Pin may have an associated AudioTrack Selector Control to indicate or select the Active AudioTrack.

The Active AudioTrack is mapped onto the Audio Effect Unit's Input Channel Configuration as described in Section 5.7.2, "Mapping Rules".

Unselected AudioTracks are passed unaltered from the incoming AVCluster to the outgoing AVCluster. Likewise, a VideoCluster or MetadataCluster present in the input AVCluster is passed unaltered to the output AVCluster.

The following figure illustrates the internal configuration of an Audio Effect Unit with respect to its AVControls.

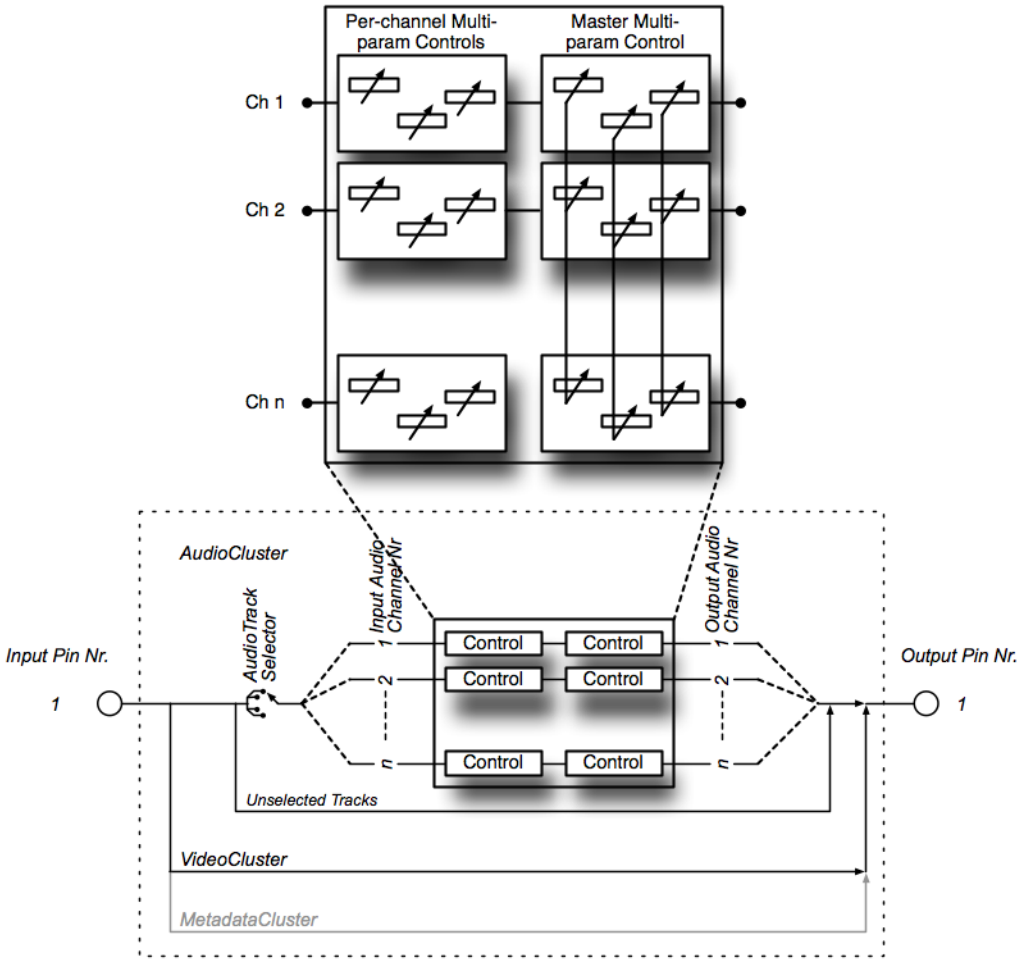


Figure 6-11: Audio Effect Unit Internal Configuration

The Audio Effect Unit shall maintain audio/video/metadata synchronization at its output at all times.

For each AudioChannel, the Audio Effect Unit provides one of the following multi-parameter AudioControls:

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

The Audio Effect Unit Descriptor reports what AVControls are present for every AudioChannel in the Audio Effect Unit. All AudioChannels in an Audio Effect Unit are fully independent. There exist no cross couplings among AudioChannels within the Unit. There are as many Output AudioChannels as there are Input AudioChannels.

The following sections describe the currently defined types of Audio Effect Units.

6.7.2.1. Parametric Equalizer Section Effect Unit

This version of the specification does not provide support for this item.

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

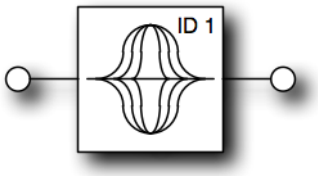


Figure 6-12: PEQS Effect Unit Icon

6.7.2.2. Reverberation Effect Unit

This version of the specification does not provide support for this item.

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

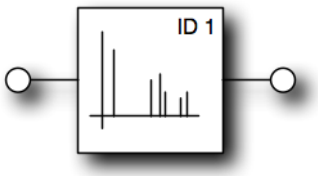


Figure 6-13: Reverberation Effect Unit Icon

6.7.2.3. Modulation Delay Effect Unit

This version of the specification does not provide support for this item.

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

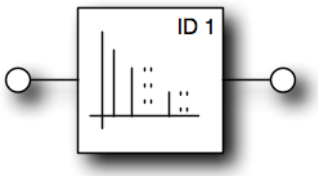


Figure 6-14: Modulation Delay Effect Unit Icon

6.7.2.4. Dynamic Range Compressor Effect Unit

This version of the specification does not provide support for this item.

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

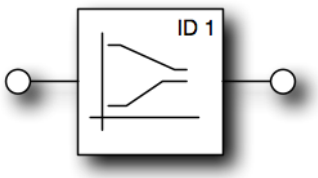


Figure 6-15: Dynamic Range Compressor Effect Unit Icon

6.8. Processing Unit

A Processing Unit (PU) is either video-only or audio-only. This specification defines several standard transforms (algorithms) that are considered necessary to support additional AV 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.

The Processing Unit shall maintain audio/video/metadata synchronization at its output at all times.

6.8.1. Video Processing Unit

The Video Processing Unit (VPU) transforms the VideoChannels of the Active VideoTrack of the incoming AVCluster into a new set of outgoing VideoChannels by applying some algorithm(s) to all the VideoChannels of the incoming Active VideoTrack as a whole.

The AVCluster enters the Video Processing Unit through a single Input Pin and leaves the Unit through a single Output Pin. The Input Pin may have an associated VideoTrack Selector Control to indicate or select the Active VideoTrack.

The Active VideoTrack is mapped onto the Video Processing Unit's Input Channel Configuration as described in Section 5.7.2, "Mapping Rules".

Unselected VideoTracks are passed unaltered from the incoming AVCluster to the outgoing AVCluster. Likewise, an AudioCluster or MetadataCluster present in the input AVCluster is passed unaltered to the output AVCluster.

The following figure illustrates the internal configuration of a Video Processing Unit with respect to its AVControls.

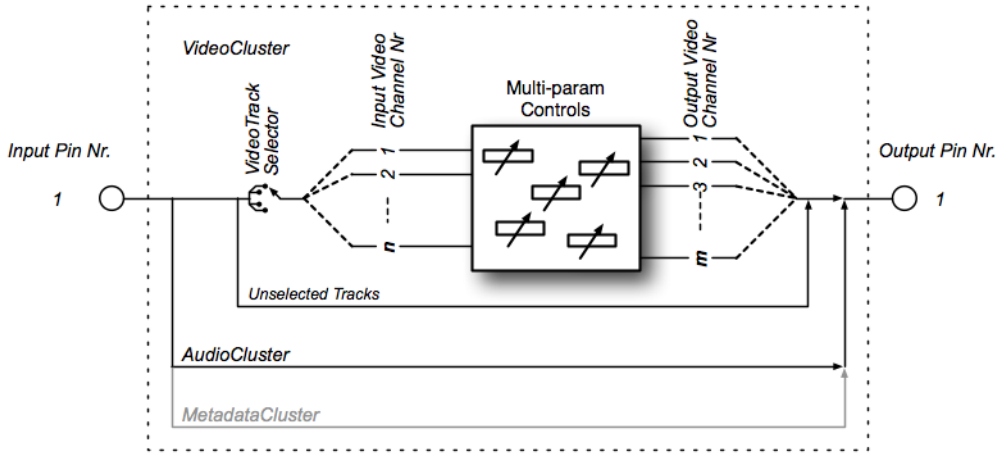


Figure 6-16: Video Processing Unit Internal Configuration

An AudioCluster or MetadataCluster present in the input AVCluster is passed unaltered to the output AVCluster.

At this time, this specification only defines the Frame Capture Video Processing Unit.

6.8.1.1. Frame Capture Processing Unit

This version of the specification does not provide support for this item.

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

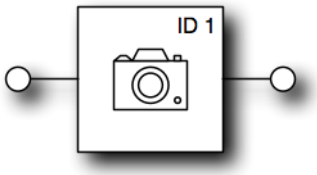


Figure 6-17: Frame Capture Processing Unit Icon

6.8.2. Audio Processing Unit

The Audio Processing Unit (APU) transforms the AudioChannels of the Active AudioTrack of the incoming AVCluster into a new set of outgoing AudioChannels by applying some algorithm(s) to all the AudioChannels of the incoming

Active AudioTrack as a whole. The AVCluster enters the Audio Processing Unit through a single Input Pin and leaves the Unit through a single Output Pin. The Input Pin may have an associated AudioTrack Selector Control to indicate or select the Active AudioTrack.

The Active AudioTrack is mapped onto the Audio Processing Unit's Input Channel Configuration as described in Section 5.7.2, "Mapping Rules".

Unselected AudioTracks are passed unaltered from the incoming AVCluster to the outgoing AVCluster. Likewise, a VideoCluster or MetadataCluster present in the input AVCluster is passed unaltered to the output AVCluster.

The following figure illustrates the internal configuration of an Audio Processing Unit with respect to its AVControls.

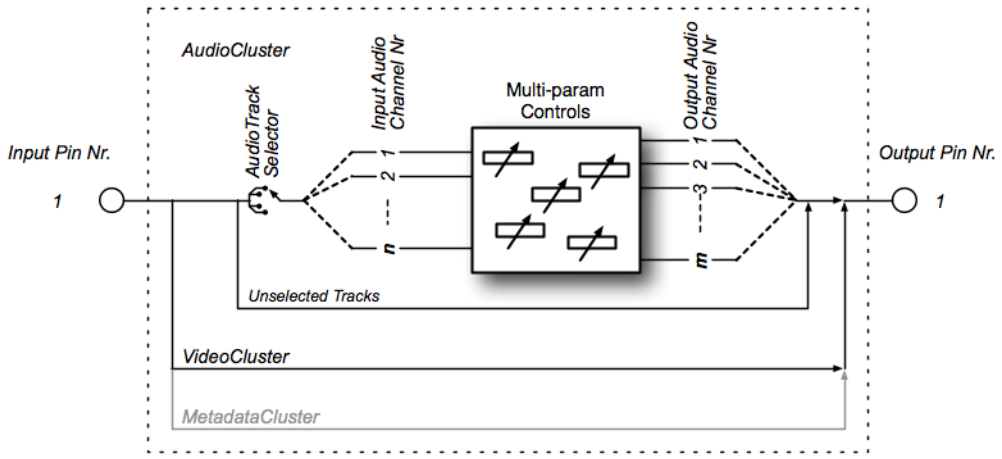


Figure 6-18: Audio Processing Unit Internal Configuration

A VideoCluster or MetadataCluster present in the input AVCluster is passed unaltered to the output AVCluster.

An Audio Processing Unit provides one of the following processing functions:

- Stereo Widening processing

6.8.2.1. Stereo Widening Processing Unit

This version of the specification does not provide support for this item.

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

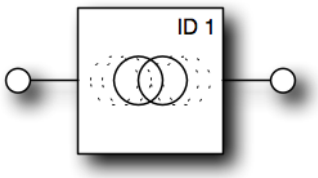


Figure 6-19: Stereo Widening Processing Unit Icon

6.9. Converter Unit

The Converter Unit (CU) provides facilities to transform the Active VideoTrack and AudioTrack of the incoming AVCluster into a VideoTrack and AudioTrack with different characteristics, such as different number of Channels, a different aspect ratio for the VideoTrack, etc. The algorithms and transforms applied to accomplish this are not defined by this specification and can be proprietary.

The AVCluster enters the Converter Unit through a single Input Pin and leaves the Unit through a single Output Pin. If the Input Pin can accept multi-track video content, then an associated VideoTrack Selector Control may be present to indicate or select the Active VideoTrack. Likewise, if the Input Pin can accept multi-track audio content, then an associated AudioTrack Selector Control may be present to indicate or select the Active AudioTrack. Unselected

VideoTracks and AudioTracks are passed unaltered from the incoming AVCluster to the outgoing AVCluster. MetadataTracks cannot be selected or processed by the Converter Unit. They too are passed unaltered from the incoming AVCluster to the outgoing AVCluster.

The Active VideoTrack and AudioTrack are mapped onto the Converter Unit's Input Channel Configuration as described in Section 5.7.2, "Mapping Rules".

Since the Converter Unit changes the incoming AVCluster into a potentially different outgoing AVCluster, the Converter Unit shall provide the following AVControl:

- Cluster (Section 9.16.1.1)

Support for any VideoControl or AudioControl listed below is optional. The Converter Unit Description reports what AVControls are present for every Channel in the Converter Unit.

The following figure illustrates the internal configuration of a Converter Unit with respect to its AVControls.

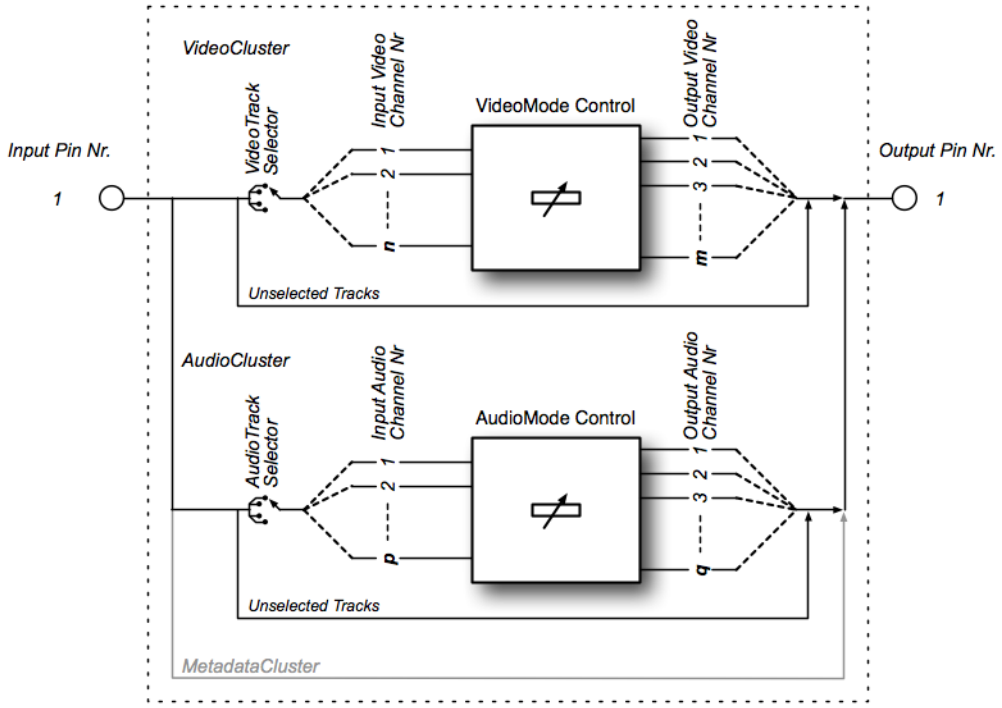


Figure 6-20: Converter Unit Internal Configuration

The Converter Unit shall maintain audio/video/metadata synchronization at its output at all times.

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

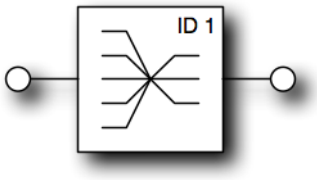


Figure 6-21: Converter Unit Icon

6.9.1. Converter Unit VideoControls

The Converter Unit redefines the configuration of the Active VideoTrack in the outgoing AVCluster by applying implementation-dependent algorithms to all or some of the available VideoChannels in the incoming Active VideoTrack, thereby potentially changing the number of VideoChannels and also their respective VideoChannel Type.

The Converter Unit provides support for the following VideoControls:

- VideoTrack Selector (Section 9.16.2.1)
- VideoMode (Section 9.16.2.2)

The Converter Unit can support multiple modes of operation. The Unit or Terminal to which the Input Pin of the Converter Unit is connected dictates the available VideoChannels in the incoming Active VideoTrack. The Converter Unit Description reports the VideoModes the Unit supports by indicating for each mode the number of VideoChannels generated in that mode and for each VideoChannel its VideoChannel Type. The aspect ratio of the resulting VideoTrack may also be altered and is therefore reported for each VideoMode as well. The VideoTrack is then mapped onto the Converter Unit's Output Channel Configuration as described in Section 5.7.2, "Mapping Rules".

As an example, consider the case where a Converter Unit is connected to an Input Terminal that represents an AVData Entity that can accept both 2D and 3D2 video content (through different AVStream Configurations). The Active VideoTrack that enters the Converter Unit therefore can contain either a single VideoChannel (OL001) or 2 VideoChannels (OL001 and OL002).

Suppose the AVFunction is designed to process 3D2 video content. Then the Converter Unit could transform incoming 2D content into 3D2 content (by duplicating the OL001 channel content into the OL002 channel or by applying some algorithm to derive 3D2 content from 2D content) and be switched in bypass mode whenever the incoming content is already 3D2.

6.9.2. Converter Unit AudioControls

The Converter Unit redefines the configuration of the Active AudioTrack in the outgoing AVCluster by applying implementation-dependent algorithms to all or some of the available AudioChannels in the incoming Active AudioTrack, thereby potentially changing the number of AudioChannels and also their respective AudioChannel Type.

The Converter Unit provides support for the following AudioControls:

- AudioTrack Selector (Section 9.16.3.1)
- AudioMode (Section 9.16.3.2)

The Converter Unit can support multiple modes of operation. The Unit or Terminal to which the Input Pin of the Converter Unit is connected dictates the available AudioChannels in the incoming Active AudioTrack. The Converter Unit Description reports the AudioModes the Unit supports by indicating for each AudioMode the number of AudioChannels generated in that mode and for each AudioChannel its AudioChannel Type. The resulting AudioTrack is mapped onto the Converter Unit's Output Channel Configuration as described in Section 5.7.2, "Mapping Rules". Note that the Converter Unit cannot alter the language associated with the Active AudioTrack.

As an example, consider the case where a Converter Unit is connected to an Input Terminal that produces Dolby AC-3 5.1 decoded audio. The Active AudioTrack that enters the Converter Unit therefore contains Front Left, Front Right, Front Center, Side Left and Side Right, and LFE AudioChannels.

Suppose the AV Function's hardware is limited to reproducing only dual channel audio. Then the Converter Unit could use some (sophisticated) algorithms to transform (down-mix) the available spatial audio information into two ('enriched') Front Left and Front Right AudioChannels so that the maximum spatial effects can be experienced, using only two AudioChannels.

As a second example, suppose the hardware is capable of servicing eight discrete AudioChannels. Now the Converter Unit could use certain techniques to derive meaningful content for the extra AudioChannels (Back Left, Back Right) that are present in the output AVCluster and are missing in the input AVCluster (AC-3 5.1). This is a typical example of an up-mix situation.

6.10. Router Unit

A Router Unit (RU) is used to combine VideoTracks, AudioTracks, and MetadataTracks available on the Input Pins of the Router Unit into one AVCluster that leaves the Router Unit via its single Output Pin.

If an Input Pin can accept multi-track video content, then an associated VideoTrack Selector Control may be present to indicate or select the Active VideoTrack. Likewise, if an Input Pin can accept multi-track audio content, then an associated AudioTrack Selector Control may be present to indicate or select the Active AudioTrack. And finally, if an Input Pin can accept multi-track metadata content, then an associated MetadataTrack Selector Control may be present to indicate or select the Active MetadataTrack.

The Router Unit provides *p* Video Input Pin Selector Controls to indicate or select the Input Pin from which the Active VideoTrack is used for each of the *p* VideoTracks in the outgoing VideoCluster. Likewise, it provides *q* Audio Input Pin Selector Controls to indicate or select the Input Pin from which the Active AudioTrack is used for each of the *q* AudioTracks in the outgoing AudioCluster. And finally, the Router Unit provides r Metadata Input Pin Selector Controls to indicate or select the Input Pin from which the Active MetadataTrack is used for each of the r MetadataTracks in the outgoing MetadataCluster.

Each of the Video/Audio/Metadata Input Pin Selector Controls has *n* possible positions where position 1 selects Input Pin 1 and position n selects Input Pin n.

The Router Unit shall maintain audio/video/metadata synchronization at its output at all times.

The following figure illustrates the internal configuration of a Router Unit with respect to its AVControls.

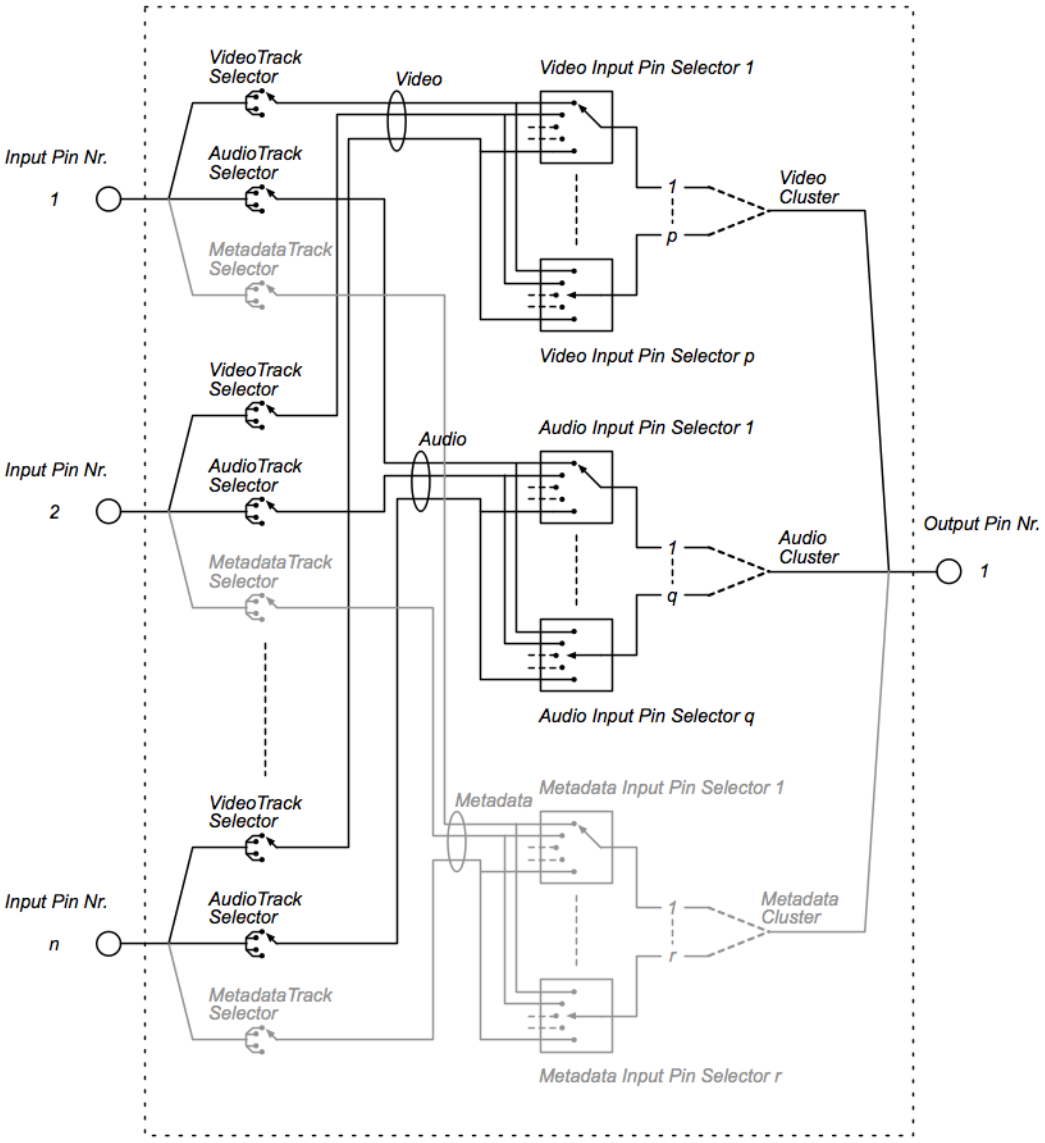


Figure 6-22: Router Unit Internal Configuration

For the simple(st) case where a Router Unit is used to combine an AVCluster that only carries video content with another AVCluster that only carries audio content, the Router Unit's internal configuration reduces to what is indicated in the following figure:

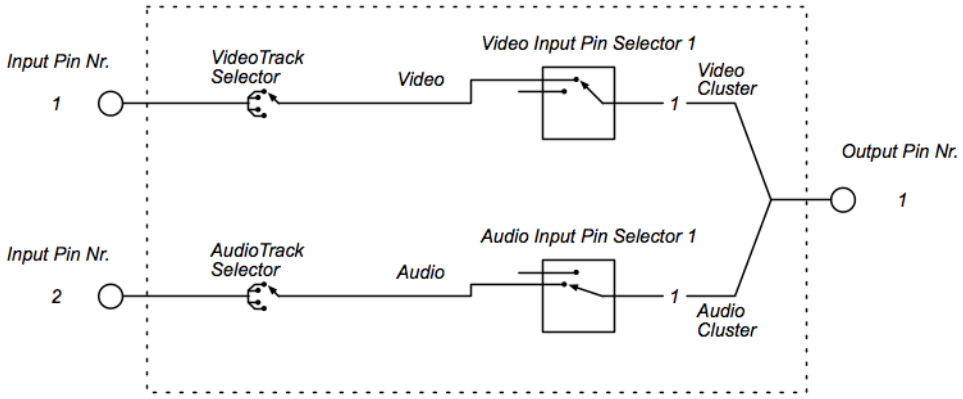


Figure 6-23: Simple Router Unit Internal Configuration Example

In this example, it is assumed that Input Pin 1 carries the video-only AVCluster and that Input Pin 2 carries the audio-only AVCluster.

The Router Unit provides support for the following AVControls:

- VideoTrack Selector (Section 9.17.1)
- AudioTrack Selector (Section 9.17.2)
- MetadataTrack Selector (Section 9.17.3)
- Video Input Pin Selector (Section 9.17.4)
- Audio Input Pin Selector (Section 9.17.5)
- Metadata Input Pin Selector (Section 9.17.6)
- Cluster (Section 9.17.7)

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

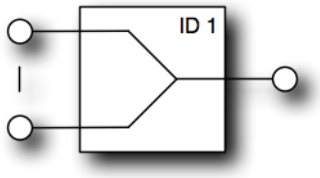


Figure 6-24: Router Unit Icon

6.10.1. Router Configurations

To describe the topology of the Router Unit, the Unit shall expose an Input Configuration for each of its Input Pins through an Input Configuration Description (ICD). Likewise, the Unit shall expose an Output Configuration for its single Output Pin through an Output Configuration Description (OCD).

An Input Configuration is characterized by a number of properties that are advertised through a corresponding Input Configuration Description:

- Video input signal path:
  - The (optional) VideoTrack Selector Control Description.
- Audio input signal path:
  - The (optional) AudioTrack Selector Control Description.
- Metadata input signal path:
  - The (optional) MetadataTrack Selector Control Description.

An Output Configuration Description contains the following information:

- Video output signal path:
  - The number of VideoTracks in the output video signal path.

- Audio output signal path:
  - The number of AudioTracks in the output audio signal path.
- Metadata output signal path:
  - The number of MetadataTracks in the output metadata signal path.

The maximum number of VideoTracks in a signal path can be up to 65,535. The maximum number of AudioTracks can also be up to 65,535 as can be the maximum number of MetadataTracks.

6.11. GraphicsEngine Entity

The GraphicsEngine Entity (GEE) is a specialized building block that represents an incorporated graphics subsystem. The output(s) generated by the graphics subsystem are available at one or more Output Pins of the GraphicsEngine Entity. The graphics subsystem inside the GraphicsEngine is controlled through the OpenGL ES graphics command language. Commands and responses are exchanged between the Controller and the GraphicsEngine via the Control Bulk Pair that is part of the AVControl Interface. In addition, the GraphicsEngine can interact with the different VideoClusters available within the AVCore. For that purpose, the GraphicsEngine has zero or more Input Pins through which selected VideoClusters can enter the GraphicsEngine and subsequently be used by the graphics subsystem for composition, texturing, etc. If an Input Pin can accept multi-track video content, then an associated VideoTrack Selector Control may be present to indicate or select the Active VideoTrack.

The Active VideoTrack is mapped onto the GraphicsEngine Entity's Input Channel Configuration as described in Section 5.7.2, "Mapping Rules" for each Input Pin.

Note that the GraphicsEngine is currently the only Entity that potentially has multiple Output Pins.

This version of the specification does not provide support for this item.

The symbol for the GraphicsEngine Entity is depicted in the following figure:

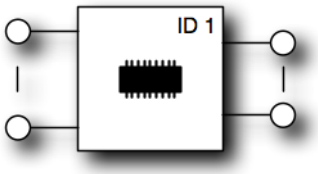


Figure 6-25: GraphicsEngine Entity Icon

6.12. AVData Entity

An AVData Entity is used to identify the exact nature of the AV source or sink the AVData Entity represents and to provide access to AVControls that are related to the production of AV content (AVData In) or the consumption of AV content (AVData Out). Each such source or sink shall be represented by an AVData Entity, even when there are no AVControls associated with that AV source or sink.

AVData Streaming Interfaces (a type of AVData Entities) describe sources and sinks that stream AV content over the USB and therefore have a separate USB interface associated with them that contains an isochronous data endpoint. An AVData Streaming Interface advertises (through its AVData Streaming Interface Description in the AVDD) exactly which AVStream Configurations it supports. Refer to the dedicated AVFormat specifications ([AVFORMAT_1], [AVFORMAT_2], and [AVFORMAT_3]) for details.

Since an AVData Entity (including the AVData Streaming Interface) represents a source or sink whose characteristics may change over time, the AVData Entity must be able to take on different characteristics as well. This is accomplished through multiple Alternate Settings of the AVData Entity, each representing a certain mode of operation for the Entity.

Note: For AVData entities that are not AVData Streaming Interfaces, the concept of an Alternate Setting is very similar to the standard USB Alternate Setting. However, class-specific methods are used to expose and select the Active Alternate Setting.

An AVData Entity cannot change its Alternate Setting by itself. It shall implement the following AVControl:

- Active Alternate Setting Control (Section 9.18.1.3)

  Note: For AVData Streaming Interfaces, the Active Alternate Setting Control shall not be implemented. Instead, the standard Get/Set Interface mechanism shall be used to manipulate the Alternate Settings of the interface.

Alternate Settings are numbered contiguously from 0 up to 255. Alternate Setting 0 is the disabled setting and shall be supported. When in Alternate Setting 0, an AVData In Entity shall cease to produce AV content to the AVCore and an AVData Out Entity shall cease to consume AV content from the AVCore. For AVData Streaming Interfaces in particular, all USB streaming to and from the interface will cease as well. An implementation may choose to switch the underlying infrastructure that instantiates the AVData Entity into a low power mode whenever Alternate Setting 0 is selected.

After Power Up or bus reset, all AVData Entities and the AVControl Interface shall assume Alternate Setting 0.

In addition to Alternate Setting 0, a least one Active Alternate Setting 1 shall be supported for any AVData Entity and the AVControl Interface.

After selection of a new configuration of the USB Device containing the AVFunction, or after the AVControl Interface is switched to or from Alternate Setting 0 all AVData Entities are required to switch to Alternate Setting 0 (without Controller intervention). Explicit Controller intervention is required to switch the AVData Entity into an Active Alternate Setting.

Note: Switching the AVControl Interface into Alternate Setting 0 effectively disables the entire AVFunction. See Section 7.2, "AVControl Interface" for details.

AVData Entities that are able to interact with AVStreams (AVData FrameBuffer Entities and AVData Streaming Interfaces) expose a list of AVStream Configurations they support in their AVData Entity Description (through the `<videoBulkStreamConfigList>`, `<videoIsoStreamConfigList>`, and `<audioStreamConfigList>` elements in the AVDD). The Controller can select the desired AVStream Configuration from the available list via the Stream Selector Control (see Section 9.18.1.6, "Stream Selector Control"). Alternatively, if the current AVStream Configuration is determined by external means, the Stream Selector Control can be Read-Only and use the Notify Message mechanism to inform the Controller what the currently selected AVStream Configuration is.

Exposing more than one Active Alternate Setting is most useful for AVData Streaming Interfaces so that the Controller can choose an appropriate bandwidth setting on the Interface that corresponds best with the selected AVStream Configuration requirements. (AVData FrameBuffer Entities will usually only expose the mandatory Alternate Setting 0 and one Active Alternate Setting 1, although a designer may choose to expose more than one Active Alternate Setting in this case as well if there is a need to be able to switch the behavior of the AVData Entity.)

It is the responsibility of the Controller to select an appropriate Alternate Setting for the AVData Entity based on the currently selected AVStream Configuration and the required USB bandwidth associated with that AVStream Configuration.

Besides the AVControls mentioned before, the following AVControls can also be provided (when applicable):

- Connectors (Section 9.18.1.1)
- Overload (Section 9.18.1.2)

An AVData Entity can advertise through its AVData Entity Description its affiliated Clock Domain and how it relates to that Clock Domain. For more information on Clock Domains, see Section 7.9, "Clock Model". When the AVData Entity acts as a Clock Domain Source, then it can provide the following AVControls (when applicable):

- Reference Clock (Section 9.18.1.7)
- Clock Connector (Section 9.18.1.8)
- Clock Valid (Section 9.18.1.9)
- Pitch (Section 9.18.1.10)

This specification supports several different types of AVData Entities as described in the following sections. Naming conventions are always using the AVCore as the point of reference. For example, an AVData Entity that delivers

content to the AVCore has the –In suffix in its name. Likewise, An AVData Entity that consumes content from the AVCore has the –Out suffix in its name. Note that this naming convention is also employed for AVData Streaming Interfaces whereas the underlying USB interfaces and endpoints use the USB naming convention where the Controller (Host) is the point of reference. Therefore an AVData Streaming-*In* interface has a USB Data *OUT* endpoint and an AVData Streaming-*Out* interface has a USB Data *IN* endpoint.

6.12.1. AVData In Entities

The AVData In Entity shall maintain audio/video/metadata synchronization when delivering content to its associated Input Terminal at all times.

The symbol for the AVData In Entity can be found in the following figure:

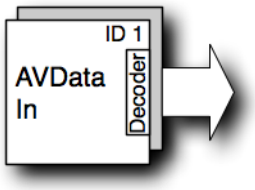


Figure 6-26: AVData In Entity Icon

6.12.1.1. AVData Generic-In Entity

This type is used to describe a generic (no specialized AVControls) AVData In Entity that accepts AV content either from an internal or an external AV source and delivers that content to the AVCore (through its associated Input Terminal). The AVData Generic-In Entity describes its intended use or function via the `<genericType:videoType>` and `<genericType:audioType>` elements in the AVData Entity Description. A list of available Video and Audio Types can be found in Appendix B, "AVData Entity Types".

Some examples are:

- Microphone
- Camera
- Radio Tuner
- TV Tuner
- …

The following AVControls can be provided (when applicable):

- Connectors (Section 9.18.2.1.1)
- Overload (Section 9.18.2.1.2)
- Active Alternate Setting (Section 9.18.2.1.3)
- Tunnel (Section 9.18.2.1.4)
- Reference Clock (Section 9.18.2.1.5)
- Clock Connector (Section 9.18.2.1.6)
- Clock Valid (Section 9.18.2.1.7)
- Pitch (Section 9.18.2.1.8)

The Connectors Control shall be implemented whenever one or more Connectors associated with the AVData Generic-In Entity provide means to detect (physically or through other means) the insertion (and removal) of a Connector.

6.12.1.2. AVData FrameBuffer-In Entity

This type is used to describe an AVData In that accepts AV content over the CBP in the form of (full or partial) frame buffer data. (See Section 7.4, "Using the CBP for Video Transport" and following for more details.)
The AVData FrameBuffer-In Entity provides the following AVControls:

- Active Alternate Setting (Section 9.18.2.2.1)
- Tunnel (Section 9.18.2.2.2)

- EDID (Section 9.18.2.2.3)
- SourceData (Section 9.18.2.2.4)
- Stream Selector (Section 9.18.2.2.5)
- Reference Clock (Section 9.18.2.2.6)
- Clock Connector (Section 9.18.2.2.7)
- Clock Valid (Section 9.18.2.2.8)

6.12.1.3. AVData Video Streaming-In Interface

This type is used to describe an AVData Entity that is accepting video content from a USB Host over the USB via a dedicated USB interface and Data OUT endpoint. See Sections 7.5, "AVData Streaming Interface" and following for more details. The AVData Video Streaming-In interface provides the following AVControls:

- (Active Alternate Setting (Section 9.18.2.3.1)) – implemented via the standard USB Set Interface request.
- Tunnel (Section 9.18.2.3.2)
- EDID (Section 9.18.2.3.3)
- Stream Selector (Section 9.18.2.3.4)
- Reference Clock (Section 9.18.2.3.5)
- Clock Connector (Section 9.18.2.3.6)
- Clock Valid (Section 9.18.2.3.7)

6.12.1.4. AVData Audio Streaming-In Interface

This type is used to describe an AVData Entity that is accepting audio content from a USB Host over the USB via a dedicated USB interface and Data OUT endpoint. See Sections 7.5, "AVData Streaming Interface" and following for more details. The AVData Audio Streaming-In interface provides the following AVControls:

- (Active Alternate Setting (Section 9.18.2.4.1)) – implemented via the standard USB Set Interface request.
- Tunnel (Section 9.18.2.4.2
- Audio Language (Section 9.18.2.4.3)
- Stream Selector (Section 9.18.2.4.4)
- Reference Clock (Section 9.18.2.4.5)
- Clock Connector (Section 9.18.2.4.6)
- Clock Valid (Section 9.18.2.4.7)
- Pitch (Section 9.18.2.4.8)

6.12.2. AVData Out Entities

The AVData Out Entity shall maintain audio/video/metadata synchronization when delivering content to its output at all times.

The symbol for the AVData Out Entity can be found in the following figure:

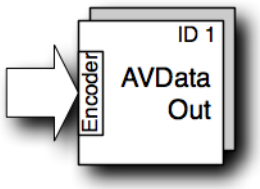


Figure 6-27: AVData Out Entity Icon

6.12.2.1. AVData Generic-Out Entity

This type is used to describe a generic (no specialized AVControls) AVData Out Entity that accepts AV content from the AVCore (through its associated Output Terminal) and delivers that content to either an internal or an external sink. The AVData Generic-Out Entity describes its intended use or function via the `<genericType:videoType>` and `<genericType:audioType>` elements in the AVData Entity Description. A list of available Video and Audio Types can be found in Appendix B, "AVData Entity Types". Some examples are:

- Display
- Speakers
- Headphones
- DP-Out (DisplayPort or Mini-DisplayPort)
- DVI-Out
- VGA-Out
- …

The following AVControls can be provided (when applicable):

- Connectors (Section 9.18.3.1.1)
- Overload (Section 9.18.3.1.2)
- Active Alternate Setting (Section 9.18.3.1.3)
- Tunnel (Section 9.18.3.1.4)
- Reference Clock (Section 9.18.3.1.5)
- Clock Connector (Section 9.18.3.1.6)
- Clock Valid (Section 9.18.3.1.7)
- Pitch (Section 9.18.3.1.8)

The Connectors Control shall be implemented whenever one or more Connectors associated with the AVData Generic-Out Entity provide means to detect (physically or through other means) the insertion (and removal) of a Connector.

6.12.2.2. AVData FrameBuffer-Out Entity

This type is used to describe an AVData Out that provides AV content over the CBP in the form of (full or partial) frame buffer data. (See Section 7.4, "Using the CBP for Video Transport" and following for more details.)
The AVData FrameBuffer-Out Entity provides the following AVControls:

- Active Alternate Setting (Section 9.18.3.2.1)
- Tunnel (Section 9.18.3.2.2)
- SinkData (Section 9.18.3.2.3)
- Stream Selector (Section 9.18.3.2.4)
- Reference Clock (Section 9.18.3.2.5)
- Clock Connector (Section 9.18.3.2.6)
- Clock Valid (Section 9.18.3.2.7)

6.12.2.3. AVData HDMI-Out Entity

This type is used to describe an HDMI output, located on the device that incorporates the AVFunction, and that receives AV content from that AVFunction. The AVData HDMI-Out Entity provides the following AVControls:

- Connector (Section 9.18.3.3.1)
- Active Alternate Setting (Section 9.18.3.3.2)
- Tunnel (Section 9.18.3.3.3)
- CEC (Section 9.18.3.3.4)
- Reference Clock (Section 9.18.3.3.59.18.3.3.69.18.3.3.7)
- Clock Connector (Section 9.18.3.3.6)
- Clock Valid (Section 9.18.3.3.7)

6.12.2.4. AVData Video Streaming-Out Interface

This type is used to describe an AVData Entity that is providing video content to a USB Host over the USB via a dedicated USB interface and Data IN endpoint. See Sections 7.5, "AVData Streaming Interface" and following for more details. The AVData Video Streaming-Out interface provides the following AVControls:

- (Active Alternate Setting (Section 9.18.3.4.1)) – implemented via the standard USB Set Interface request.
- Tunnel (Section 9.18.3.4.2)
- Stream Selector (Section 9.18.3.4.3)

- Reference Clock (Section 9.18.3.4.4)
- Clock Connector (Section 9.18.3.4.5)
- Clock Valid (Section 9.18.3.4.6)

6.12.2.5. AVData Audio Streaming-Out Interface

This type is used to describe an AVData Entity that is providing audio content to a USB Host over the USB via a dedicated USB interface and Data IN endpoint. See Sections 7.5, "AVData Streaming Interface" and following for more details. The AVData Audio Streaming-Out interface provides the following AVControls:

- (Active Alternate Setting (Section 9.18.3.5.1)) – implemented via the standard USB Set Interface request.
- Tunnel (Section 9.18.3.5.2)
- Stream Selector (Section 9.18.3.5.3)
- Reference Clock (Section 9.18.3.5.4)
- Clock Connector (Section 9.18.3.5.5)
- Clock Valid (Section 9.18.3.5.6)
- Pitch (Section 9.18.3.5.7)

6.13. Encoders and Decoders

Encoders and decoders are not stand-alone Entities as such. Instead they are always embedded within an AVData Entity.

6.13.1. Decoders

The conversion process from incoming – possibly encoded – AVStreams to logical Channels always involves some kind of decoding engine. This specification defines several types of decoding processes, ranging from rather trivial decoding schemes like converting interleaved stereo 16 bit PCM audio data into a Front Left and Front Right AudioChannel to very sophisticated schemes like converting an MPEG-4 encoded AVStream into one or more VideoChannels, a number of AudioChannels, and potentially some MetadataChannels as well. The decoding engine is always present as part of the AVData In Entity.

The decoding processes each have their specific Decoder Controls defined that allow manipulation or monitoring of the internal status of the decoder process. Control Sequences specific to the decoding engine shall be directed to the AVData In Entity.

At this time, this specification does not include support for controlling decoders.

6.13.2. Encoders

The conversion process from outgoing logical Channels to possibly encoded AVStreams always involves some kind of encoding engine. This specification defines several types of encoding processes, ranging from rather trivial to very sophisticated schemes. An encoding engine is always present as part of the AVData Out Entity.

The encoding processes each have their specific Encoder Controls defined that allow manipulation or monitoring of the internal status of the encoder process. Control Sequences specific to the encoding engine shall be directed to the AVData Out Entity.

At this time, this specification does not include support for controlling encoders.

7. Operational Model

A USB 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 AVFunctions can exist in the same device. Interfaces, belonging to the same AVFunction are grouped into an AV Interface Association. If the device contains multiple independent AVFunctions, there shall be multiple AV Interface Associations, each providing full access to their associated AVFunction.

As an example of a composite device, consider a PC monitor equipped with a built-in stereo speaker system. Such a device would possibly have two interfaces to deal with its AV aspects. One of those, the AVControl Interface, is used to control the inner workings of the function (Volume Control, Brightness Control, etc.) and also to convey frame buffer video updates to its AVData FrameBuffer-In Entity via the CBP whereas the other, the AVData Audio Streaming Interface, handles the audio data traffic, sent to the monitor's audio subsystem.

The AVData Audio Streaming Interface could be configured to operate in different modes through the selection of different AudioStream Configurations and Alternate Settings, if necessary, due to USB bandwidth considerations, for that interface. One AudioStream Configuration may support PCM stereo audio whereas another may support 5.1 encoded audio data.

From an interface point of view, such a setup requires one isochronous endpoint in the AVData Audio Streaming Interface to receive the compressed or uncompressed audio stream, in addition to the mandatory control endpoint and Control Bulk Pair endpoints in the AVControl Interface.

If the above AVData Audio Streaming Interface were an asynchronous sink, one extra isochronous Feedback endpoint would also be necessary.

As stated earlier, AV functionality is located at the interface level in the device class hierarchy. The following sections describe the AV Interface Association, containing a single AVControl Interface and optional AVData Streaming Interfaces, together with their associated endpoints that are used for AVFunction control and for AVStream transfer.

7.1. AV Interface Association

On the USB, an AVFunction is completely defined by its interfaces. An AVFunction has one AVControl Interface and zero or more AVData Streaming Interfaces, grouped into an AV Interface Association. The standard USB Interface Association Descriptor (IAD) shall be used to describe the AV Interface Association binding those interfaces together, even if there is only one interface in the Association. A device is allowed to have multiple AV Interface Associations active at the same time. These Associations are then used to control multiple, independent AVFunctions located in the same composite device. Interface Association is expressed via the standard USB Interface Association Descriptor (IAD). Every Interface Association Descriptor has a FunctionClass, FunctionSubClass and FunctionProtocol field that together identify the function that is represented by the Association. The following paragraphs define these fields for the AV Device Class.

7.1.1. AVFunction Class

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

The AVFunction Class code is assigned by this specification. For details, see Appendix A.1, "AVFunction Class Code".

7.1.2. AVFunction Subclass

The AVFunction Class is divided into Function Subclasses. At this moment, the Function Subclass code is not used and shall be set to FUNCTION_SUBCLASS_UNDEFINED.

The assigned Subclass codes can be found in Appendix A.2, "AVFunction Subclass Codes" of this specification. All other Subclass codes are unused and reserved by this specification for future use.

7.1.3. AVFunction Protocol

The AVFunction Class and Subclasses can be further qualified by the Function Protocol code. The Function Protocol code is is not used and shall be set to FUNCTION_PROTOCOL_UNDEFINED.

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

7.1.4. AV Interface Class

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

For an AVFunction to be compliant to this specification, the only requirement is that it exposes one AVControl Interface. No further interaction with the AVFunction is mandatory, although AVFunctions in the AV Class may support one or more optional AVData Streaming Interfaces for consuming or producing one or more media data streams.

The AV Interface class code is assigned by the USB-IF. For details, see Appendix A.4, "AV Interface Class Code".

7.1.5. AV Interface Subclass

The AV Interface class is divided into Subclasses. All AVFunctions are part of a certain Interface Subclass. The following Interface Subclasses are currently defined in this specification:

- AVControl Interface Subclass
- AVData Video Streaming Interface Subclass
- AVData Audio Streaming Interface Subclass

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

7.1.6. AV Interface Protocol

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

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

7.2. AVControl Interface

To control the functional behavior of a particular AV Function, the Controller can manipulate the AVData Entities, Units, and Terminals inside the AVFunction. To make these objects accessible, the AVFunction shall expose a single AVControl Interface. This interface shall contain the following endpoints:

- The control endpoint 0 (the default pipe) for manipulating the standard USB aspects of the interface. All standard requests, such as Set Interface etc. are sent over this endpoint. This endpoint is a shared resource among all interfaces that constitute the (composite) USB Device.
- One Bulk IN and one Bulk OUT endpoint. Together they constitute the Control Bulk Pair (CBP) that is used for all class-specific interaction between the Controller and the AVFunction and therefore, the CBP shall be present for every Alternate Setting of the AVControl Interface. All AVControls present in the AVFunction are addressed through this Control Bulk Pair. Also, all notifications sent to the Controller make use of the CBP. Finally, the Controller also retrieves the AV Description Document through the CBP.

The AVControl Interface is the single entry point to access the internals of the AVFunction. All Control Sequences (see further) shall be directed to the AVControl Interface of the AVFunction.

The AVControl Interface of an AVFunction shall support at least two Alternate Settings and can potentially support multiple Alternate Settings, each representing a particular configuration of the AVFunction.

Alternate Setting 0 shall be supported and represents the disabled state of the entire AVFunction (including the AVControl Interface). When Alternate Setting 0 is activated, the AVFunction ceases all operation. All other AVData Streaming Interfaces belonging to the same Interface Association shall also cease all operation and return to Alternate Setting 0. The entire AVFunction can now potentially enter a low power state.

Besides Alternate Setting 0, at least one Active Alternate Setting 1 shall be supported for the AVControl Interface. Additional Alternate Settings may be supported.

When the Controller selects an Active Alternate Setting for the AVControl Interface, the AVFunction shall become active and shall present the same state as if the USB Device containing the AVFunction would have been reset or powered up. All AVData Entities in the AVFunction, including all AVData Streaming Interfaces, shall start in Alternate Setting 0 and therefore require explicit Controller intervention to be set to an Active Alternate Setting.

7.2.1. Control Endpoint

The AV interface class uses endpoint 0 (the default pipe) as the standard way to control the standard USB aspects of its Interfaces (AVControl and AVData Streaming). The standard requests that are supported by this specification are further detailed in Section 9.1, "Standard Requests" of this document.

7.2.2. Control Bulk Pair Endpoints

The AVControl Interface shall contain one Bulk IN and one Bulk OUT endpoint that together constitute the Control Bulk Pair (CBP). The AV class uses the Control Bulk Pair to convey class-specific Control Sequences to and from the AVFunction. These Control Sequences are always directed to or originating from a specific AVControl inside the AVFunction. The format and contents of these Control Sequences are detailed further in Section 9.2, "Class-Specific AVControl Sequences" of this document.

7.3. Control Bulk Pair (Theory of Operation)

Note: This specification currently only describes the operation of the CBP in the case where the Controller functionality is located in the USB Host. The following sections should be interpreted with this restriction in mind. Future versions of this specification may address the case where the Controller functionality is located in the USB Device.

Control Sequences are transported on the CBP according to specific requirements:

- Improve bus efficiency
- Reduce Host and Device processing overhead
- Allow the Controller and Controllee to reduce power usage

There are specific requirements for use of the Bulk OUT and Bulk IN pipes of the CBP. These requirements are described in the following sections and define how Command, Response, Notify and Null messages are transported on the CBP Bulk Pipes.

7.3.1. CBP Bulk Transfers

There are several requirements that are the same for both CBP Bulk OUT and CBP Bulk IN transfers.

The Host is the transmitter of Messages on the CBP Bulk OUT pipe and the AVFunction is the transmitter of Messages on the CBP Bulk IN pipe. The phrase "CBP transmitter" is used to generically refer to the Host and AVFunction when they are the transmitter of Messages on a CBP Bulk Pipe. The phrase "CBP receiver" is used to generically refer to the Host and AVFunction when they are the receiver of Messages on a CBP Bulk Pipe.

A CBP Bulk transfer is physically comprised of one or more Messages, or in the case of a Zero-Length Packet (ZLP) no Messages.

The CBP transmitter combines as many Messages in a single transfer as it desires. Combining multiple Messages in a single transfer increases bus efficiency for the transfer. However, every Message is timely and it is recommended that implementations not delay the transfer of Messages by waiting for an additional Message to become available.

Since each Message is always padded to a multiple of AV_MESSAGE_GRANULARITY, the beginning of a Message in a data payload will always be aligned on AV_MESSAGE_GRANULARITY byte boundaries.

Messages are transported on a CBP Bulk pipe using the minimum number of data payloads that are required given the total size of the Messages in a transfer. For small Messages, multiple Messages can be transported in a single data payload. For large Messages, a single Message may require several data payloads. A single Message may span two or more data payloads. A single Message may span two or more transfers.

Each CBP Bulk data payload contains one or more Messages or a part of a Message.

Messages in a transfer are transported in data payloads in Message order where the end of one Message is immediately followed by the beginning of the next Message.

Since Messages may be contained in an integer multiple of data payloads, the CBP transmitter must have a method to inform the CBP receiver that a transfer of a group of Messages is complete and should be processed. Otherwise, a transfer that is smaller than the buffering provided by the CBP receiver could remain as an uncompleted transfer and the CBP receiver would never process the Messages in that transfer. The normal USB short packet semantic is used as the method to ensure that the CBP receiver's transfers complete.

When the CBP transmitter has partially filled a data payload in preparation for sending it to the CBP receiver and the CBP transmitter has no more Messages currently available, the CBP transmitter shall queue the partial payload for transmission to the CBP receiver (Short Packet).

When a CBP transmitter has sent a wMaxPacketsize sized data payload to the CBP receiver as the previous data payload and has no more Messages currently available, the CBP transmitter shall queue a data payload that contains only a Null Message for transmission to the CBP receiver. This ensures that all transfers are terminated by a short packet (a Null Message is considerably smaller than wMaxPacketSize).

These two possible actions (partial data payload and Null Message data payload) ensure that the transport path adds minimal latency to Message processing.

Figure 7-1 shows 4 examples of the four patterns for how Messages are transported in data payloads for a CPB Bulk transfer. Example 1 shows a single Message, containing less than wMaxPacketsize bytes. This message is transported using short packet semantics. Example 2 shows a Message that is exactly wMaxPacketSize bytes in length. In this case a Null Message is queued by the CBP transmitter to terminate the transfer. The third example shows multiple Messages, concatenated into several data payloads where the last data payload is less than wMaxPacketSize bytes in length. Again, short packet semantics are used to terminate the transfer. Lastly, the fourth example shows the case where the end of the last Message coincides with a wMaxPacketSize boundary. Again, in this case a Null Message is queued by the CBP transmitter to terminate the transfer.

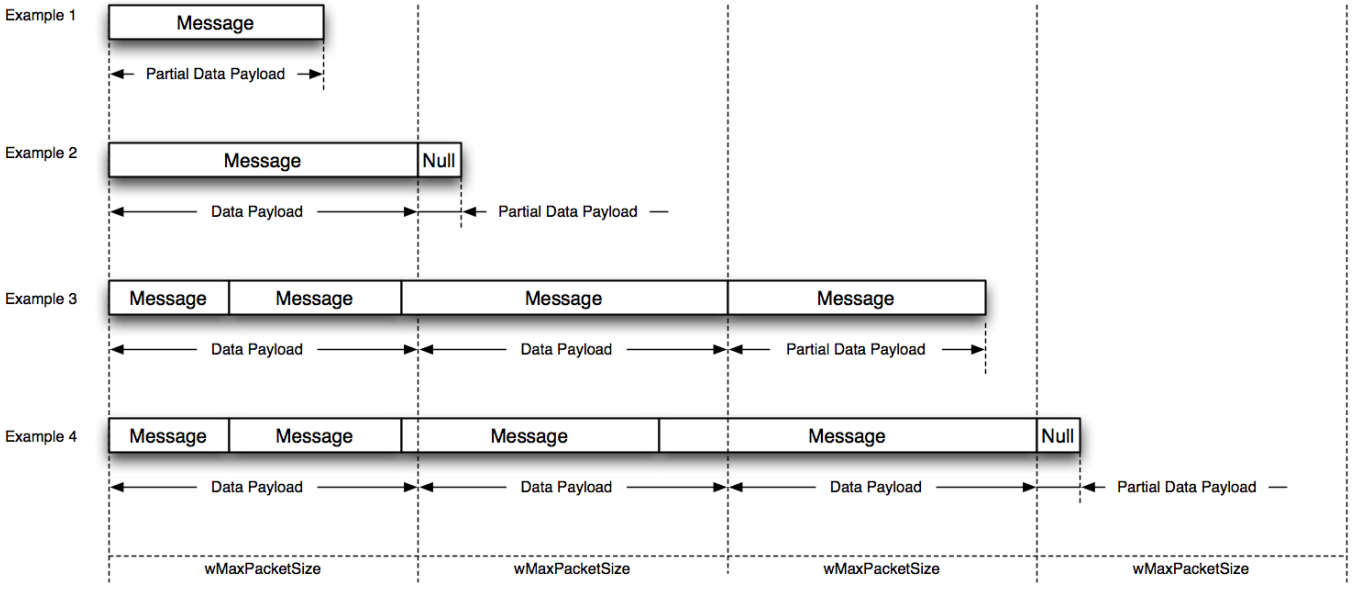


Figure 7-1: CBP Bulk Messages and Data Payloads

7.3.1.1. High-Speed CBP Bulk Flow Control

AVFunctions operating at High-Speed have no additional flow control mechanism.

7.3.1.2. SuperSpeed CBP Bulk Flow Control

AVFunctions operating at SuperSpeed shall use normal NRDY-ERDY protocol for flow control, when data for a data payload is not available.

7.3.1.3. CBP Bulk Message Combining Recommendations

USB efficiency is maximized for a CBP Bulk Pipe when most Messages are transmitted to the CBP receiver in the fewest data payloads. However, holding Messages that could be transmitted until there are sufficient Messages to fill a complete transfer request can unnecessarily delay the transmission and processing of those Messages. Therefore, it is important to combine as many messages as possible for transmission, without compromising minimum latency.

The following recommendations presume there is some ability of the CBP transmitter to detect that there are transfers awaiting transmission to the CBP receiver. The precision of this ability is implementation specific and outside the scope of this specification. The following recommendations use the phrase “transfer opportunities are available” to mean that there are no (or at most a few) transfers awaiting transmission to the CBP receiver. The phrase “transfer opportunities are not available” is used to mean that the CBP transmitter has one or more transfers that are awaiting transmission to the CBP receiver.

When the CBP transmitter determines to send one or more Messages to the CBP receiver and transfer opportunities are available, the CBP transmitter should queue the Messages for transmission. Note that this transfer will either end with a partial data payload due to the Messages being transmitted, or a Null Message (partial payload) will also be added at the end of the transfer.

When the CBP transmitter determines to send one or more Messages to the CBP receiver and transfer opportunities are not available, the CBP transmitter should accumulate the Message(s) until transfer opportunities become available. When one (or more) transfer(s) complete, the CBP transmitter should queue the Messages for transmission.

7.3.2. CBP Bulk IN Transfers

The following are specific requirements for the CBP Bulk IN pipe.

The Host shall maintain a single IN transfer request pending for the CBP Bulk IN pipe. CBP Bulk IN transfers will transport Response, Notify, and Null Messages as well as Zero-Length Packets (ZLPs). A CBP Bulk IN transfer request shall have buffering set at some integer multiple of wMaxPacketSize such that it will accept an appropriate number of data payloads, determined based on the responses expected from the AVFunction.

7.3.3. CBP Bulk IN Idle

AVFunctions operating at Full- or High-Speed can encounter situations where the Host is continually issuing IN tokens asking for data from the AVFunction, but the AVFunction has no data to provide so these continuous IN transactions are completed with NAKs. To avoid this situation the Full- or High-Speed connection uses a CBP Idle Condition where the Host will stop issuing unnecessary IN tokens and allow the bus to be used more efficiently for other (non-CBP) purposes.

The Full- or High-Speed Host will enter the CBP Idle Condition whenever there are no outstanding Response Messages and exit the CBP Idle Condition whenever an OUT transfer has been queued.

To avoid the problem of blocking Notify Messages at the AVFunction when there is significant time between OUT transfers, the Full- or High-Speed Host can receive Notify Messages or Zero Length Packets from the CBP Bulk IN pipe in response to requested IN transfers following any system timer event that occurs during a CBP Idle Condition.

There is no CBP Idle Condition for AVFunctions operating at SuperSpeed. The SuperSpeed Bulk IN flow control is sufficient.

7.3.3.1. Host CBP Bulk IN Idle Behavior

When a transfer request by a Host completes for the CBP Bulk IN pipe, the Host will issue another transfer request so that there is an IN transfer request pending for the pipe so long as the count of Response Messages due is greater than zero. A CBP Idle Condition occurs when the number of Response Messages reaches zero.

To avoid the problem of Notify Messages being held for long periods of time at the AVFunction, it is assumed that there is a continuous system timer event that occurs at approximately a 50 or 60 Hz rate (20 or 16 ms approximately).

When the system timer event has occurred and there are currently transfers awaiting transmission, the Host should take no special action in response to the system timer event. When the system timer event has occurred and there are no transfers awaiting transmission, the Host should request a transfer of a Null Command Message to the AVFunction on the CBP Bulk OUT Pipe and ensure that an IN transfer request is pending on the CBP Bulk IN pipe.

If the Host receives a transfer with non-zero length, it should ensure that an IN transfer request is pending on the CBP Bulk IN pipe.

Figure 7-2 contains a state machine that shows the Host CBP Bulk IN Idle behavior. The CBP Idle Condition state means that the Host has no pending IN transfers; the Response Msg Bulk IN Transfer Pending state means that the Host has a pending IN transfer and is expecting Response Messages but will also transport Notify Messages and ignore ZLP transfers; the ZLP Bulk IN Transfer Pending state means that the Host has a pending IN transfer and is expecting a zero length transfer while allowing any Notify Messages to be transported.

Note that a CBP Bulk IN transfer may contain zero, one or many Response Messages such that a single IN transfer completion may carry all the outstanding Response Messages. Whenever an IN transfer completes and the count of outstanding Response Messages is greater than zero the Host queues another CBP Bulk IN transfer request.

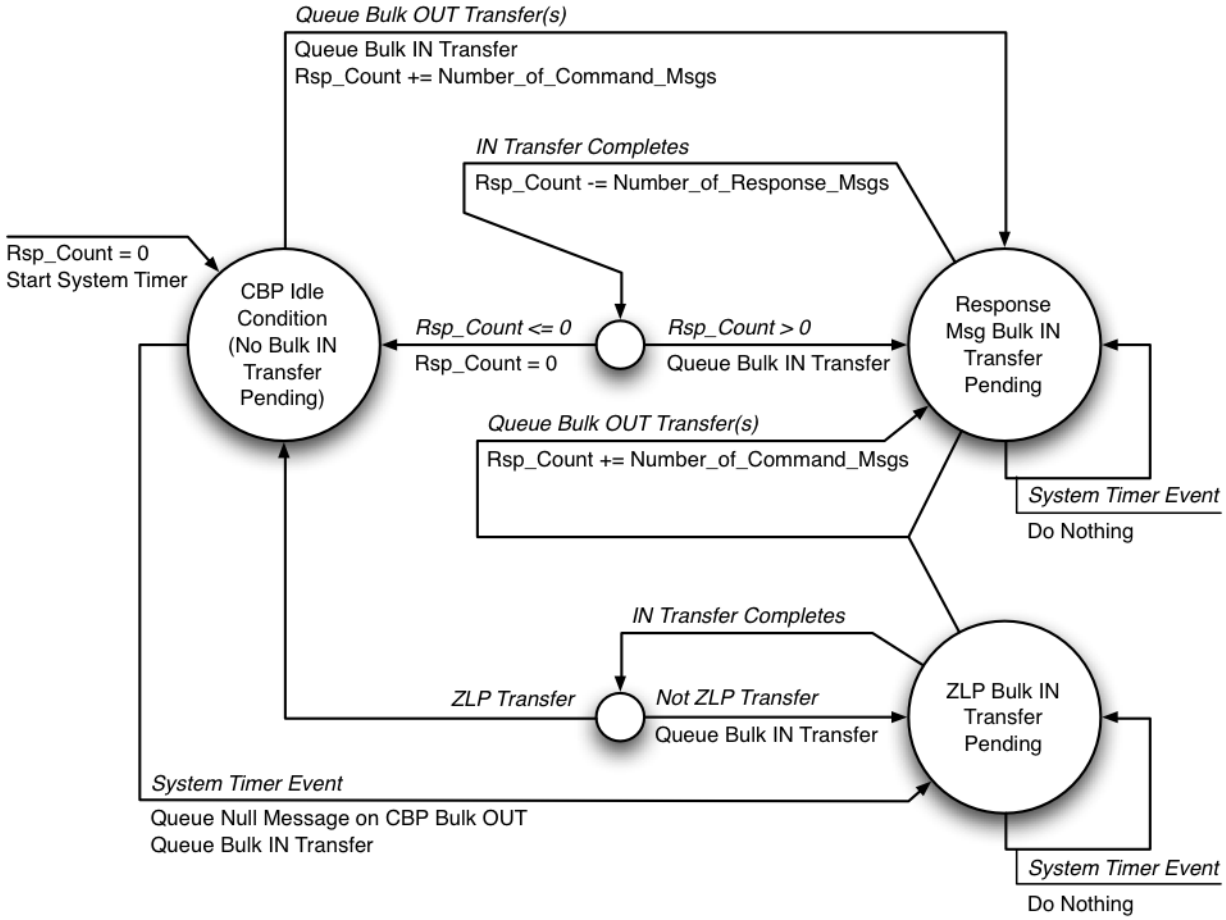


Figure 7-2: Host CBP Bulk IN Idle Behavior

7.3.3.2. AVFunction CBP Bulk IN Idle Behavior

When an AVFunction receives a CBP Bulk OUT transfer, it shall determine the number of Command Messages and add that count to a local Rsp_Msg_Count store. When the AVFunction determines to queue a Bulk IN transfer it shall

determine the number of Response Messages in the transfer (corresponding to each Command Message completed) and subtract that count from the local Rsp_Msg_Count store.

When an AVFunction has a Notify Message to send, it shall queue the Notify Message (following as appropriate the CBP Bulk Message Combining recommendations).

When the AVFunction receives on the CBP Bulk OUT Pipe a data payload consisting of only a Null Command Message, the AVFunction shall queue a Zero Length Packet to the Host on the CBP Bulk IN Pipe.

Figure 7-3 contains a state machine that shows the AVFunction CBP Bulk IN Idle behavior. The No Response Messages Pending state corresponds to the CBP Bulk Idle Condition and means that the AVFunction has no Host Command Messages to process; the Response Messages Pending state means that the AVFunction has Host Command Messages to process.

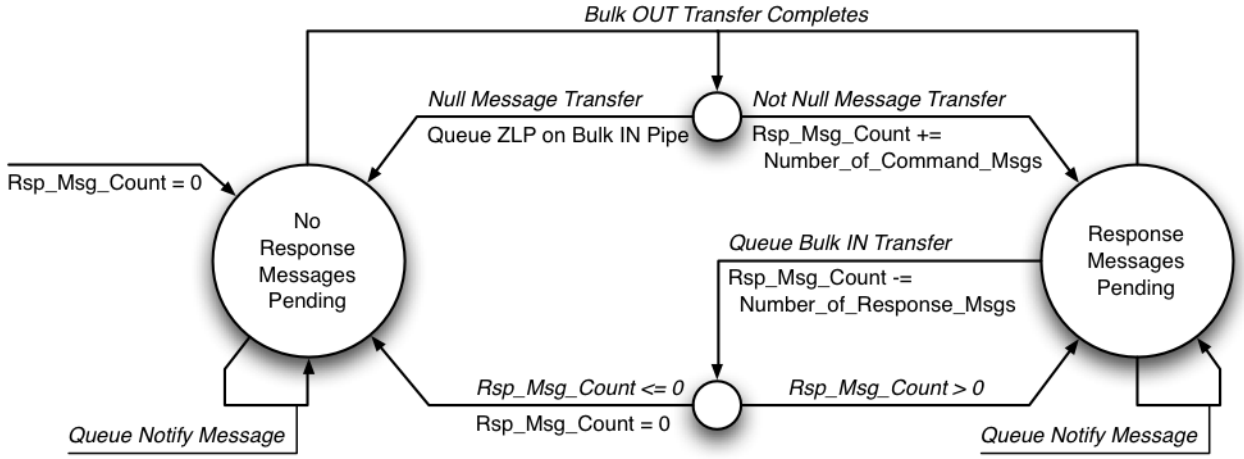


Figure 7-3: AVFunction CBP Bulk IN Idle Behavior

7.4. Using the CBP for Video Transport

Video content can be transported either over the CBP (Video-Only over Bulk) or via a dedicated AVData Video Streaming Interface. The following Sections describe the use of the CBP and the various Entities involved in the transport of video content into and out of the AVFunction.

7.4.1. Delivering Video Content to the AVFunction

The Controller delivers video content to the AVFunction through one or more AVData FrameBuffer-In Entities within the AVFunction. Each AVData FrameBuffer-In Entity contains a SourceData AVControl. The SourceData AVControl acts as a framebuffer and the AVData FrameBuffer-In Entity essentially delivers the current content of its SourceData AVControl as video frames through its associated Input Terminal to the internals of the AVCore. The Controller addresses the SourceData AVControl via a regular SETI Control Sequence and the data payload of the SETI Control Sequence contains the new video content that is used to update the SourceData framebuffer. The video content can contain either a (partial) VideoFrame update or can contain a compressed VideoFrame update, encapsulated in a specific Message structure. For details about the exact format of the Messages a SourceData AVControl can accept, refer to [AVFORMAT_1]. For details on supported VideoStream Configurations and how to select them, refer to Section 7.8, “AVStream Advertising and Selection”.

7.4.2. Retrieving Video Content from the AVFunction

The Controller retrieves video content from the AVFunction through one or more AVData FrameBuffer-Out Entities within the AVFunction. Each AVData FrameBuffer-Out Entity contains a SinkData AVControl. The SinkData AVControl acts as a framebuffer and the AVData FrameBuffer-Out Entity essentially receives VideoFrames through its associated Output Terminal from the internals of the AVCore and exposes this content through its SinkData AVControl as subsequent VideoFrame updates. The Controller addresses the SinkData AVControl via a regular GET Control Sequence and the data payload of the GET Control Sequence contains the new video content that is retrieved from the SinkData framebuffer. The video content can contain either a (partial) VideoFrame update or can contain a

compressed VideoFrame update, encapsulated in a specific Message structure. For details about the exact format of the Messages a SinkData AVControl can produce, refer to [AVFORMAT_1]. For details on supported VideoStream Configurations and how to select them, refer to Section 7.8, "AVStream Advertising and Selection".

7.5. AVData Streaming Interface

AVData Streaming Interfaces are a type of AVData Entities that are used to interchange digital AVStreams between the Controller and the AVFunction. They are optional. An AVFunction can have zero or more AVData Streaming Interfaces associated with it, each possibly carrying data of a different nature and format. Each AVData Streaming Interface can have at most one data endpoint. This construction guarantees a one-to-one relationship between the AVData Streaming Interface and the single AVStream, related to the endpoint, and may be 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 [USB2.0] or [USB3.0], Section 9.6.6, "Endpoint."

An AVData Streaming Interface can have Alternate Settings that can be used to change certain characteristics of the interface and underlying endpoint. An AVData Streaming Interface shall at least provide Alternate Setting 0 (disabled) with zero bandwidth requirements (no isochronous data endpoint defined) and one Active Alternate Setting 1 that contains the actual isochronous data endpoint. Additional Alternate Settings may be supported.

For every Data OUT or IN endpoint defined in any of the AVData Streaming Interfaces, there shall be a corresponding Input or Output Terminal defined in the AVCore. For the Controller to fully understand the nature and behavior of the connection, it should take into account the interface- and endpoint-related descriptors and Descriptions as well as the Terminal-related Descriptions.

7.5.1. Synchronization Types

Each isochronous AV endpoint used in an AVData Streaming Interface belongs to a synchronization type as defined in Section 5 of [USB2.0] and Section 4 of [USB3.0]. The following sections briefly describe the possible synchronization types.

7.5.1.1. Asynchronous

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

7.5.1.2. Synchronous

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

7.5.1.3. Adaptive

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

7.6. AVData Video Streaming Interface

Video content can be transported either over the CBP (Video-Only over Bulk compressed or partial update video streaming uses the CBP and the AVData FrameBuffer Entity mechanisms as decribed in Section 7.4, "Using the CBP for Video Transport") or via a dedicated AVData Video Streaming Interface. The following Sections describe the AVData Video Streaming Interface and its components.

7.6.1. Isochronous AVData Video Endpoint

In general, the streams that are handled by an isochronous AVData Video endpoint do not necessarily map directly to the logical VideoChannels that exist within the AVCore. As an example, consider the case where multiple video channels are compressed into a single data stream (MVC, …). The format of such a data stream can be entirely different from the native format of the video channels. Therefore, to describe the data transfer at the endpoint level correctly, the notion of VideoCluster (logical VideoChannels and VideoTracks) is replaced by the notion of a

VideoBundle, which is very similar in structure to a VideoCluster but with all the physical characteristics of the VideoStream preserved. It is the responsibility of the decoder inside the AVData Video Streaming-In interface to 'convert' between the VideoBundle and the VideoCluster before handing the data over to the Input Terminal. Likewise, the encoder inside the AVData Video Streaming-Out interface shall convert the VideoCluster from the Output Terminal into a VideoBundle. If the decoder or encoder exposes AVControls that influence the decoding or encoding process, these AVControls can be accessed through the AVControl Interface.

An AVData Video Streaming Interface can have zero (for Alternate Setting 0) or one isochronous AVData Video endpoint. If multiple tightly related (from a content perspective) synchronous video channels are communicated between Controller and AVFunction, they should be bundled into one VideoBundle, and the result can be directed to the single endpoint.

If an AVFunction needs more than one VideoBundle to operate, each VideoBundle shall be directed to the endpoint of a separate AVData Video Streaming Interface, belonging to the same AV Interface Association (all servicing the same AVFunction).

7.6.2. Isochronous Feedback Endpoint

In general, for adaptive video source endpoints and asynchronous video sink endpoints, an explicit synchronization mechanism is needed to maintain synchronization during transfers. For details about synchronization, see Section 5, "USB Data Flow Model," in [USB2.0] or Section 4, "SuperSpeed Data Flow Model" in [USB3.0]. For specific information on the formatting of the data going over the feedback pipe, refer to Section 5.12.4.2, "Feedback" and Section 9.6.6, "Endpoint" in [USB2.0] or Section 4.4.8.4.1, "Explicit Feedback" in [USB3.0].

7.6.3. VideoStreams Configurations

The configurations used to transport VideoStreams over the USB for a particular AVData Video Streaming Interface is entirely described in the AVDD section that provides a detailed XML description of that interface. For each Alternate Setting of that interface, all the necessary information is provided for the Controller to determine exactly which VideoStream Configurations can be used.

At this time, this specification only supports uncompressed full-frame video streaming over an AVData Video Streaming Interface. For details about the defined VideoStream Configurations and Descriptions, see the separate document [AVFORMAT_3] that is considered part of this specification.

7.6.4. Inter Channel Synchronization

When dealing with multi-channel video, it is important to keep all VideoChannels synchronized when these VideoChannels are flowing through the AVFunction. An AVFunction shall therefore maintain strict channel synchronization for all VideoChannels in an AVCluster.

7.7. AVData Audio Streaming Interface

This version of the specification only considers audio content being streamed over an AVData Audio Streaming Interface. The following Sections describe the AVData Audio Streaming Interface and its components.

7.7.1. Isochronous AVData Audio Endpoint

In general, the streams that are handled by an isochronous AVData Audio endpoint do not necessarily map directly to the logical AudioChannels that exist within the AVCore. As an example, consider the case where multiple 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 audio channels (for example, 640 Kbits/s AC-3 5.1 audio as opposed to 6 channel 16 bit 44.1 kHz PCM audio). Therefore, to describe the data transfer at the endpoint level correctly, the notion of AudioCluster (logical AudioChannels and AudioTracks) is replaced by the notion of an AudioBundle, which is very similar in structure to an AudioCluster but with all the physical characteristics of the AudioStream stream preserved. It is the responsibility of the decoder inside the AVData Audio Streaming-In interface to 'convert' between the AudioBundle and the AudioCluster before handing the data over to the Input Terminal. Likewise, the encoder inside the AVData Audio Streaming-Out interface shall convert the AudioCluster from the Output Terminal into an AudioBundle. If the decoder or encoder exposes AVControls that influence the decoding or encoding process, these AVControls can be accessed through the AVControl Interface.

As already mentioned, an AVData Audio Streaming Interface can have zero or one isochronous AVData Audio endpoint. If multiple tightly related (from a content perspective) synchronous audio channels are communicated between Controller and AVFunction, they should be bundled into one AVBundle, and the result can be directed to the single endpoint.

If an AVFunction needs more than one AudioBundle to operate, each AudioBundle shall be directed to the endpoint of a separate AVData Audio Streaming Interface, belonging to the same AV Interface Association (all servicing the same AVFunction).

7.7.2. Isochronous Feedback Endpoint

For adaptive audio source endpoints and asynchronous audio sink endpoints, an explicit synchronization mechanism is needed to maintain synchronization during transfers. For details about synchronization, see Section 5, "USB Data Flow Model," in [USB2.0]. For specific information on the formatting of the data going over the feedback pipe, refer to Section 5.12.4.2, "Feedback" and Section 9.6.6, "Endpoint" in the USB Specification.

7.7.3. AudioStream Configurations

The configurations used to transport AudioStreams over the USB for a particular AVData Audio Streaming Interface is entirely described in the AVDD section that provides a detailed XML description of that interface. For each Alternate Setting of that interface, all the necessary information is provided for the Controller to determine exactly which AudioStream Configurations can be used.

For details about the defined AudioStream Configurations and Descriptions, see the separate document [AVFORMAT_2] that is considered part of this specification.

7.7.4. Inter Channel Synchronization

An important issue when dealing with audio, and 3-D audio in particular, is the phase relationship between different AudioChannels. 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 AudioChannels used to reproduce the audio source. Therefore, it is imperative that USB AVFunctions respect the phase relationship among all related AudioChannels when these AudioChannels are flowing through the AVFunction. An AVFunction shall therefore maintain strict channel synchronization for all AudioChannels in an AVCluster.

7.8. AVStream Advertising and Selection

In the current version of the specification, video content can be exchanged between the Controller and the AVFunction via two different types of AVData Entities: the AVData FrameBuffer Entity and the AVData Video Streaming Interface. Audio content is exchanged exclusively via the AVData Audio Streaming Interface, a type of AVData Entity.

As previously described, each AVData Entity (including the AVData Video or Audio Streaming Interfaces) shall support at least two Alternate Settings: Alternate Setting 0 (disabled) and one or more Active Alternate Settings. The AVData Entity advertises in its AVDD Description the list of AVStream (VideoStream or AudioStream) Configurations it supports. In addition, the AVData Entity contains a Stream Selector Control that allows the Controller to either select the AVStream Configuration it wants to use at a particular moment in time or to determine which AVStream Configuration was selected by the AVData Entity through external means (see Section 9.18.1.6, "Stream Selector Control" for more details).

Each AVStream Configuration is assigned a one-based index value, based on the order in which the AVStream Configuration appears in the list of advertised AVStream Configurations in the AVDD Description. The Stream Selector Control uses this index value to indicate or select a particular configuration.

Changing the AVStream Configuration for an AVData Entity always involves a reconfiguration of the underlying infrastructure (hardware or software). Therefore, changing the AVStream Configuration shall always involve a cycle through Alternate Setting 0 (disabled) of the AVData Entity. The procedure is as follows:

For Read-Write Stream Selector Control:

- The Controller stops the ongoing AVStream in the current Active Alternate Setting by switching the AVData Entity to Alternate Setting 0.

For an AVData In Entity (Controller streams to the AVData In Entity), the Controller should first cease the streaming process to that AVData In Entity before switching to Alternate Setting 0.

- The Controller then selects the desired AVStream Configuration.
- The Controller finally selects the Alternate Setting that provides the necessary bandwidth for the selected AVStream Configuration.
  For an AVData Out Entity, this is an indication that it may start streaming to the Controller.

For Read-Only Stream Selector Control:

- Upon reception of the Stream Selector Control Notify Message (that contains the new desired AVStream Configuration), the Controller stops the ongoing AVStream in the current Active Alternate Setting by switching the AVData Entity to Alternate Setting 0.
  For an AVData In Entity (Controller streams to the AVData In Entity), the Controller should first cease the streaming process to that AVData In Entity before switching to Alternate Setting 0.
- The Controller then selects the Alternate Setting that provides the necessary bandwidth for the new desired AVStream Configuration.
  For an AVData Out Entity, this is an indication that it may start streaming to the Controller.

When an AVData Entity supports multiple Video- or AudioStream Configurations, it is up to the designer of the AVFunction to decide how many different Alternate Settings to expose.

For AVData FrameBuffer Entities, there are no reserved USB bandwidth considerations and one Active Alternate Setting will usually suffice, although a designer may have other criteria that would require multiple Active Alternate Settings.

For AVData Streaming Interfaces, this decision could be guided by the USB bandwidth each of the supported AVStream Configurations requires. It is recommended that the Alternate Settings be designed in such a way that their reserved USB bandwidth covers the required USB bandwidth of one or more of the supported AVStream Configurations. At a minimum, there shall be at least one Active Alternate Setting available with enough reserved USB bandwidth for each supported AVStream Configuration.

7.9. Clock Model

As stated before, inside the AVCore, AV functionality is described through an abstract logical model and most physical characteristics of the AVClusters flowing through the AVCore are considered irrelevant.

Outside the AVCore however, in the realm of the AVData Entities and AVStreams, those physical characteristics do become relevant.

AVStreams are characterized by a number of parameters. For VideoStreams, the number of VideoChannels in a VideoTrack, the number of VideoTracks in a VideoBundle, the VideoFrame Rate, the VideoFrame and VideoSample Formats are all relevant parameters. For AudioStreams, the number of AudioChannels in and AudioTrack, the number of AudioTracks in an AudioBundle, the AudioFrame Rate, and the AudioSample Format are equally important.

AVData FrameBuffer Entities and AVData Streaming Interfaces that exchange these AVStreams with the Controller are characterized by a similar set of parameters, advertising the different AVStream Configurations they support. The characteristics of an AVStream that is exchanged with an AVData Entity shall match one of the advertised AVStream Configurations for streaming to successfully occur.

One important parameter in that matching process is the operating frequency (Video- or AudioFrame Rate) at which the actual streaming is going to occur.

Every AVData Entity inside the AVFunction is implicitly or explicitly affiliated to a particular Clock Domain (see Section 7.9.1, “Clock Domains”). Ultimately, it is the Clock Domain Reference Clock that determines the operating frequency of that Entity, either directly ($operating\ frequency = Reference\ Clock\ frequency$) or indirectly ($operating\ frequency = Reference\ Clock\ frequency * factor$).

For AVData Streaming Interfaces, it is required to express Clock Domain affiliation through the AVData Streaming Interface Description. For AVData Entities other than AVData Streaming Interfaces, expressing Clock Domain affiliation is optional. However, even for those Entities, it may be necessary in some cases to include Clock Domain

Descriptions to correctly indicate to Controller software that certain clock-related bindings exist between AVData Entities. For example, this could be used to express that an AVData FrameBuffer Entity is affiliated to the same Clock Domain as an AVData Audio Streaming Interface so that the VideoStream and the AudioStream for those two Entities are synchronous in nature. As another example, a certain AVData Entity may provide the Reference Clock for a Clock Domain (possibly through an external connector) without necessarily generating an AVStream by itself.

Note: This specification does not provide explicit tools to manage the synchronization between Bulk transport-based AVStreams and Isochronous-based AVStreams.

All video-related frequency values used in this specification shall tolerate at least ±5000 PPM inaccuracy to accommodate for sampling clock inaccuracies. Likewise, all audio-related frequency values shall tolerate at least ±1000 PPM.

7.9.1. Clock Domains

A Clock Domain is defined as a zone within which all AVData Entities that are connected to that Clock Domain use the same Reference Clock to generate their local sampling clocks (either video or audio clocks). There is always one and only one AVData Entity that shall advertise itself as the Clock Domain Source. Any other AVData Entity that is affiliated to that Clock Domain shall advertise itself as a Clock Domain Sink. In other words, a Clock Domain shall always have a single Clock Domain Source and may have zero or more Clock Domain Sinks. All AVData Entities that are affiliated with the same Clock Domain are by definition synchronous in nature. An AVData Entity can only be affiliated to a single Clock Domain (when an AVData Entity manipulates Video and Audio simultaneously, the Video- and AudioStreams are always synchronous).

The `<clockDomain>` element in the AVData Entity Description indicates how the AVData Entity is related to the particular Clock Domain. It expresses whether the AVData Entity is a Clock Domain Source or Sink and, in the case of a Clock Domain Source, indicates the presence of the optional Reference Clock, Clock Valid and Pitch Controls.

Note: A unique Clock Domain ID identifies each Clock Domain.

In addition, for AVData Streaming Interfaces, the Controller should query the standard isochronous data endpoint descriptor to discover the USB synchronization type of the Clock Domain (asynchronous/synchronous/adaptive).

The Clock Domain Reference Clock does not have to be valid at all times. For instance, if an AVData Entity is the Clock Domain Source and the Clock Domain Reference Clock is derived from the average incoming data rate, the Reference Clock might not be valid when the AVData Entity is currently not receiving any data. The optional Clock Valid Control (may be present in an AVData Entity Alternate Setting that acts as a Clock Domain Source) can indicate the validity of the Reference Clock at all times.

In general, multiple (different) Clock Domains can exist within the same AVFunction.

Inside the AVCore, AVClusters can be bridged from one Clock Domain to another through the use of internal rate conversion processes. The sampling rate conversion processes are not exposed to the Controller since the Controller does not need to interact with them. The fact that an AVCluster originates from an Input Terminal connected to a certain Clock Domain (via its associated AVData Entity) and, after being processed by the AVCore, leaves the AVCore through an Output Terminal that is connected to a different Clock Domain (again, via its associated AVData Entity) is an indicator that the AVCluster must have undergone some sort of sampling rate conversion in between.

7.9.2. Clock Domains and AVData Streaming Interfaces

Each AVData Streaming Interface shall expose to the Controller its affiliated Clock Domain and how it relates to that Clock Domain via the AVData Streaming Interface Description. Selection of the operating frequency of the interface happens indirectly by selecting a particular AVStream Configuration, which includes the VideoFrame or AudioFrame Rate parameter. Selecting an AVStream Configuration is accomplished either by the Controller programming the Stream Selector Control of the AVStreaming Interface to the desired AVStream Configuration or by the AVFunction selecting a particular AVStream Configuration and notifying the Controller of the selection through a Stream Selector Notify Message. In all cases, it is the the responsibility of the Controller to select the appropriate Alternate Setting of the AVData Streaming Interface that supports the USB bandwidth required by the selected AVStream Configuration as described in Section 7.8, “AVStream Advertising and Selection”.

Note: To keep the number of Active Alternate Settings in an AVData Streaming Interface to a minimum, it is not recommended to provide a separate Active Alternate Setting for every combination of operating frequency, resolution, etc. Instead, it is probably enough to support just a few Active Alternate Settings (low, medium, and high bandwith) to provide reasonable bandwidth control.

7.10. Binding between Physical Buttons and AVControls

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

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 AVControl

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

7.10.1. Physical button is a HID Control

This case is referred to as remote control and the physical button is completely separate from the AVFunction and is implemented within the device's HID interface. The AVFunction is not even aware of the button's existence. Any change of state for the button is communicated to Controller software via HID reports. It is then up to Controller software to interpret the button state change and derive from there the appropriate action to be taken toward the AVFunction. 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.

7.10.2. Physical button is Integral Part of the AVControl

This case is referred to as local control and the physical button directly interacts with the actual AVControl. Button state changes are not reported to Controller software via HID reports. Instead, the change of state of the AVControl resulting from the button manipulation is reported to Controller software through the AVControl notification mechanism. As a consequence, the binding between the physical button and the AVControl 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.

7.11. Use of Legacy View

During normal USB enumeration, the Device that contains the AVFunction is required to return a set of Legacy View Descriptors as described in Section 8.3, "Legacy View Descriptors". This allows the Controller to interact with (potentially reduced) AV functionality on the Device without having to read and parse the entire XML AV Description Document (AVDD). However, in most cases, the AVDD will return much more detailed information than what is comprised in the Legacy View Descriptors. In fact, some functionality may not even be described through the Legacy View Descriptors and is only discoverable via the AVDD. Such functionality shall however not interfere with the Legacy View operation of the Device. In other words, that functionality shall assume default behavior and not require interaction from the Controller in order for the AVFunction to operate properly, albeit sometimes with reduced functionality.

Note that Legacy View operation is not a special operational mode of the AVFunction. Instead, it is a way in which the Controller interacts with (parts of) the AVFunction as decribed by a set of Descriptors rather than through the AVDD.

8. AV Descriptors and the AV Description Document

The AVFunction advertises its standard USB functionality and characteristics through a set of standard USB descriptors.

In general, the AV Class does not use USB descriptors to convey class-specific information. Instead, all class-specific information that an AVFunction provides to the Controller is bundled into a single XML Document, called the AV Description Document (AVDD).

However, to allow Controller to interoperate (most likely, at a reduced functionality level) with the AVFunction when the Controller does not have a full-featured AV Class driver available (during the boot process, for example), an AVFunction shall expose (possibly reduced) AV class-specific information through a set of Legacy View Descriptors that provide enough information to the Controller to interact with the AVFunction to a certain operational level without having to parse the entire AVDD. For details, refer to Section 8.3, "Legacy View Descriptors".

8.1. Standard USB AV Descriptors

An AVFunction is always defined at the interface level and often coexists with other functionality (for example HID). Therefore, in most cases, the USB Device in which the AV functionality is implemented is a composite USB Device. The standard Device, Configuration, Interface, and Endpoint Descriptors that such a Device shall expose are defined in [USB2.0] and [USB3.0] and the USB Device that incorporates the AV Functionality shall adhere to the rules and descriptor layouts as defined by those specifications.

It is highly recommended that USB Devices that incorporate AV functionality use the value triplet (0xEF, 0x02, 0x01) for their **bDeviceClass**, **bDeviceSubclass**, and **bDeviceProtocol** fields in the standard Device Descriptor (see [USBIADDCC]).

8.2. Class-specific AV Description Document (AVDD)

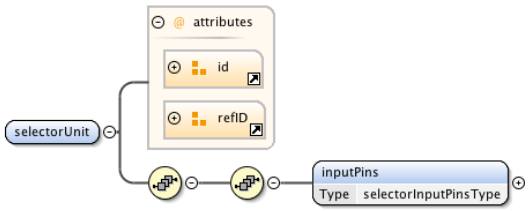
All class-specific descriptive information about the AVFunction is retrieved from the Device via the AV Description Document (AVDD). The AVDD is an XML Document that contains a highly structured and hierarchical description of all the building blocks (Entities) that together constitute the AVFunction. For each Entity, the AVDD contains all the necessary information for the Controller to determine which AVControls are present in the Entity and what the operating conditions are for each of these AVControls. Furthermore, the AVDD contains topological information on how all the Entities are interconnected so that Controller software can determine all the available signal paths within the AVFunction.

The Controller retrieves the AVDD from the AVFunction through the AVDD Info Control and the AVDD Content Control (See Sections 9.8.1, "AVDD Controls".) The AVDD Info and AVDD Content Controls shall always be present in the AVControl Interface (EntityID=1) and are accessible to the Controller at all times.

The AVDD XML Document shall conform to the AVSchema, which can be found at [AVSCHEMA]. It must be noted that many of the requirements imposed by this specification are reflected in the syntactical constructs as defined in the AVSchema but some requirements cannot be (adequately) expressed using the XML Schema syntax. Therefore, to build a compliant AVFunction, a designer shall not solely rely on the AVSchema but take all the requirements defined in this specification into consideration. Whenever there is a discrepancy between requirements imposed by the AVSchema and requirements defined in this specification, this specification takes precedence.

All Entity Descriptions are documented via the AVSchema Documentation. The AVSchema contains annotations that clarify the use of all elements, attributes and their type definitions. The AVSchema Documentation is automatically generated from the AVSchema. It consists of a set of html files that together constitute a fully hyperlinked and browsable mini website. The following is a documentation example, describing a typical schema component (in this case, the `<selectorUnit>` element).

Element selectorUnit

Namespace	http://www.usb.org/schemas/AVSchema
Annotations	Describes a Selector Unit.
Diagram	
Properties	Content: **complex**
Used by	Element **avControlInterface/alternateSetting**
Model	**inputPins**
Children	**inputPins**
Instance	


```
<selectorUnit id="" refID="">
  <inputPins>{1,1}</inputPins>
</selectorUnit>
```

Attributes

QName	Type	Fixed	Default	Use	Annotation
id	**idType**			required	Uniquely identifies the Entity represented by the element within the AVFunction. The value of the id attribute shall be unique for each Entity within the AVFunction. The value 1 is reserved for the Control interface Entity. The value of this attribute shall be used in the EID parameter of any Control Sequence that addresses this Entity.
refID	**xsi:string**			optional	Can be used to uniquely identify the object represented by this element for external binding reference purposes. The value of the refID shall be unique for each object within the AVFunction and is implementation-specific. May be a GUID.

Source

```
<xsi:element name="selectorUnit">
  <xsi:annotation>
    <xsi:documentation>Describes a Selector Unit.</xsi:documentation>
  </xsi:annotation>
  <xsi:complexType>
    <xsi:sequence>
      <xsi:sequence>
        <xsi:element name="inputPins" type="selectorInputPinsType"/>
      </xsi:sequence>
    </xsi:sequence>
    <xsi:attributeGroup ref="id"/>
    <xsi:attributeGroup ref="refID"/>
  </xsi:complexType>
</xsi:element>
```

Schema location: file:/Users/Geert/Documents/Data/PROJECTS/XML/AVSchema/AVSchema.xsd

Figure 8-1: AVSchema Documentation Example

The information is presented in the form of a table. The first column lists the available information items. Whenever applicable, names and diagram components are active hyperlinks to more detailed underlying information.

The Instance item presents a pseudo-XML Fragment instantiation of the schema component under consideration. Implementation-dependent element values are evidently not available from the schema. Instead, the element value is replaced by the notation `{x,y}` where x represents the `minOccurs` value for the element and y represents the `maxOccurs` value for the element. Likewise, implementation-dependent attribute values are not available and are replaced by the empty string (“”).

The Source item presents the actual schema definition of the component.

8.2.1. Configuration Description

The root element for all compliant AV Description Documents shall be `<avConfiguration>`. This root element encapsulates the entire AVFunction Description.

The AV Configuration Description Documentation can be found at: AV Configuration.

8.2.2. AVControl Interface Description
The AVControl Interface Description Documentation can be found at: AVControl Interface.

8.2.3. Input Terminal Description
The Input Terminal Description Documentation can be found at: Input Terminal.

8.2.4. Output Terminal Description
The Output Terminal Description Documentation can be found at: Output Terminal.

8.2.5. Mixer Unit Description
The Mixer Unit Description Documentation can be found at: Mixer Unit.

8.2.6. Selector Unit Description
The Selector Unit Description Documentation can be found at: Selector Unit.

8.2.7. Feature Unit Description
The Feature Unit Description Documentation can be found at: Feature Unit.

8.2.8. Converter Unit Description
The Converter Unit Description Documentation can be found at: Converter Unit.

8.2.9. Router Unit Description
The Router Unit Description Documentation can be found at: Router Unit.

8.2.10. AVData Entities

8.2.10.1. AVData Generic-In Entity Description
The AVData Generic-In Entity Description Documentation can be found at: AVData Generic-In Entity.

8.2.10.2. AVData Generic-Out Entity Description
The AVData Generic-Out Entity Description Documentation can be found at: AVData Generic-Out Entity.

8.2.10.3. AVData FrameBuffer-In Entity Description
The AVData FrameBuffer-In Entity Description Documentation can be found at: AVData FrameBuffer-In Entity.

8.2.10.4. AVData FrameBuffer-Out Entity Description
The AVData FrameBuffer-Out Entity Description Documentation can be found at: AVData FrameBuffer-Out Entity.

8.2.10.5. AVData HDMI-Out Entity Description
The AVData HDMI-Out Entity Description Documentation can be found at: AVData HDMI-Out Entity.

8.2.10.6. AVData Video Streaming-In Entity Description
The AVData Video Streaming-In Entity Description Documentation can be found at: AVData Video Streaming-In Interface.

8.2.10.7. AVData Video Streaming-Out Entity Description
The AVData Video Streaming-Out Entity Description Documentation can be found at: AVData Video Streaming-Out interface.

8.2.10.8. AVData Audio Streaming-In Entity Description
The AVData Audio Streaming-In Entity Description Documentation can be found at: AVData Audio Streaming-In Interface.

8.2.10.9. AVData Audio Streaming-Out Entity Description

The AVData Audio Streaming-Out Entity Description Documentation can be found at: AVData Audio Streaming-Out interface.

8.2.10.10. Clock Domain Description

The Clock Domain Description Documentation can be found at: Clock Domain Description.

8.3. Legacy View Descriptors

The Legacy View Descriptors describe Entities, AVControls, and AVStream Configurations supported by an AVFunction operating in Legacy View. Legacy View is usually (but not necessarily) defined as a subset of the AVFunction's full functionality and provides a reduced set of AV capabilities, sufficient to allow the Controller to use the AVFunction for example, during its boot sequence. The Legacy View Descriptors describe the AV Class-specific aspects of an AVFunction operating in Legacy View.

The AV Legacy View Descriptors defined in this specification may only be modified or extended by revision of this specification.

8.3.1. Relationship between Legacy View Descriptors and the AV Description Document

The full AVFunction functionality is described through the AVDD, an XML Document that fully documents all the features and characteristics of the AVFunction (see Section 4.3, "AV Description Document" for details). Every AVFunction shall expose its full functionality through a hierarchical structure of feature Descriptions (see Section 8.2, "Class-specific AV Description Document (AVDD)"). Each Description is an XML Fragment that describes one particular feature or aspect of the AVFunction's functionality. It also expresses the hierarchical position of that feature within the overall AVDD structure. For example, the Volume Control Description is part of the Feature Unit Description to which the Volume Control belongs. In turn, the Feature Unit Description is part of the AVControl Interface Description to which the Feature Unit belongs. Finally, the AVControl Interface Description is part of the AVConfiguration Description that provides the highest level of hierarchy, encompassing the entire class-specific Description of the AVFunction. The XML AVSchema ([AVSCHEMA]) governs the syntax and hierarchy of the AVDD.

The AVFunction shall also expose a set of feature Legacy View Descrip*tors*, very similar to the feature Descrip*tions* contained in the AVDD, but usually only describing a subset of the AVFunction's functionality. Each Legacy View Descriptor provides the same information about the feature as can be found in the Description, but expressed in a different format and sometimes only advertising limited information about the feature.

The AVDD Descriptions use XML whereas the Legacy View Descriptors use the standard USB Descriptor syntax.

The AVDD is required to describe and expose each and every feature or aspect of the AVFunction so that the Controller can have a full understanding of the inner workings of the AVFunction by parsing the AVDD XML Document. In contrast, the set of Legacy View Descriptors may expose only a limited set of features of the AVFunction, just enough to allow the Controller to operate the AVFunction in a meaningful fashion. All features and aspects of the AVFunction that are not explicitly exposed through Legacy View Descriptors shall assume default behavior after the AVControl Interface is switched from Alternate Setting 0 to an Active Alternate Setting and shall be constructed in such a way that no Controller intervention is required with those features or aspects in order for the AVFunction to provide the functionality as exposed through the Legacy View Descriptors only.

As a rule of thumb, only the mandatory features and aspects of the AV Device Class Definition will be supported through the Legacy View Descriptors (although this is not a requirement). For AVControls, only Master Controls can be supported (there is no provision to indicate the existence of AVControls on a per-channel basis).

8.3.2. Legacy View Descriptor Hierarchy

The Legacy View Descriptors are returned as part of the Configuration Descriptor set that a USB Device returns. The Legacy View Descriptors follow the standard USB interface Descriptors for each of the interfaces that are part of the AVFunction.

The AVFunction is composed of one AVControl Interface and zero or more AVData Streaming Interfaces. The AVControl Interface and each of the AVData Streaming Interfaces (if any) are represented by standard USB Interface Descriptors in the traditional hierarchy of USB Descriptors that together constitute the overall Configuration

Descriptor of the USB Device. The total length (in bytes) of the Configuration Descriptor (with all of its constituent parts: standard, class-specific, and vendor-specific) is reported in the **wTotalLength** field of the standard Configuration Descriptor.

Since all of the Descriptors start with a **bLength** field, followed by a **bDescriptorType** field, a Controller should have no problem correctly parsing the set of standard, class-specific and vendor-specific Descriptors that make up the total Configuration Descriptor. The **bLength** field indicates the length of the Descriptor, in bytes. The **bDescriptorType** field indicates the type of the Descriptor. Refer to A.13.1, "Descriptor Type Codes" for a list of available Descriptor Types.

The class-specific Legacy View Descriptors shall be hierarchically interspersed with the standard USB Descriptors. Specifically, all class-specific AVControl Interface-related Descriptors shall follow the standard USB AVControl Interface Descriptor. Likewise, all class-specific AVData Streaming Interface-related Descriptors shall follow their standard USB AVData Streaming Interface Descriptor counterparts.

Furthermore, the order in which the class-specific Legacy View Descriptors for a given Interface appear is important and shall follow the same ordering rules, as are in place for their Description equivalents. In other words, the ordering of the Legacy View Descriptors for a specific Interface is governed by the ordering rules imposed by the AVSchema on their equivalent feature Descriptions.

The following figure illustrates the hierarchy of the Legacy View Descriptors (example only). Not all Descriptors depicted in the figure need to be present in a given implementation. Only those Descriptors relevant to the implementation shall be present. The notation $^{1}2_{3}$ is used to indicate the possible occurrence of multiple instances of the marked item. For example, the AVControl Interface Descriptor may be followed by multiple occurrences of a Terminal Descriptor, each of which may be followed by multiple occurrences of an AVControl Descriptor, each of which may be followed by multiple occurrences of a Ranges Descriptor.

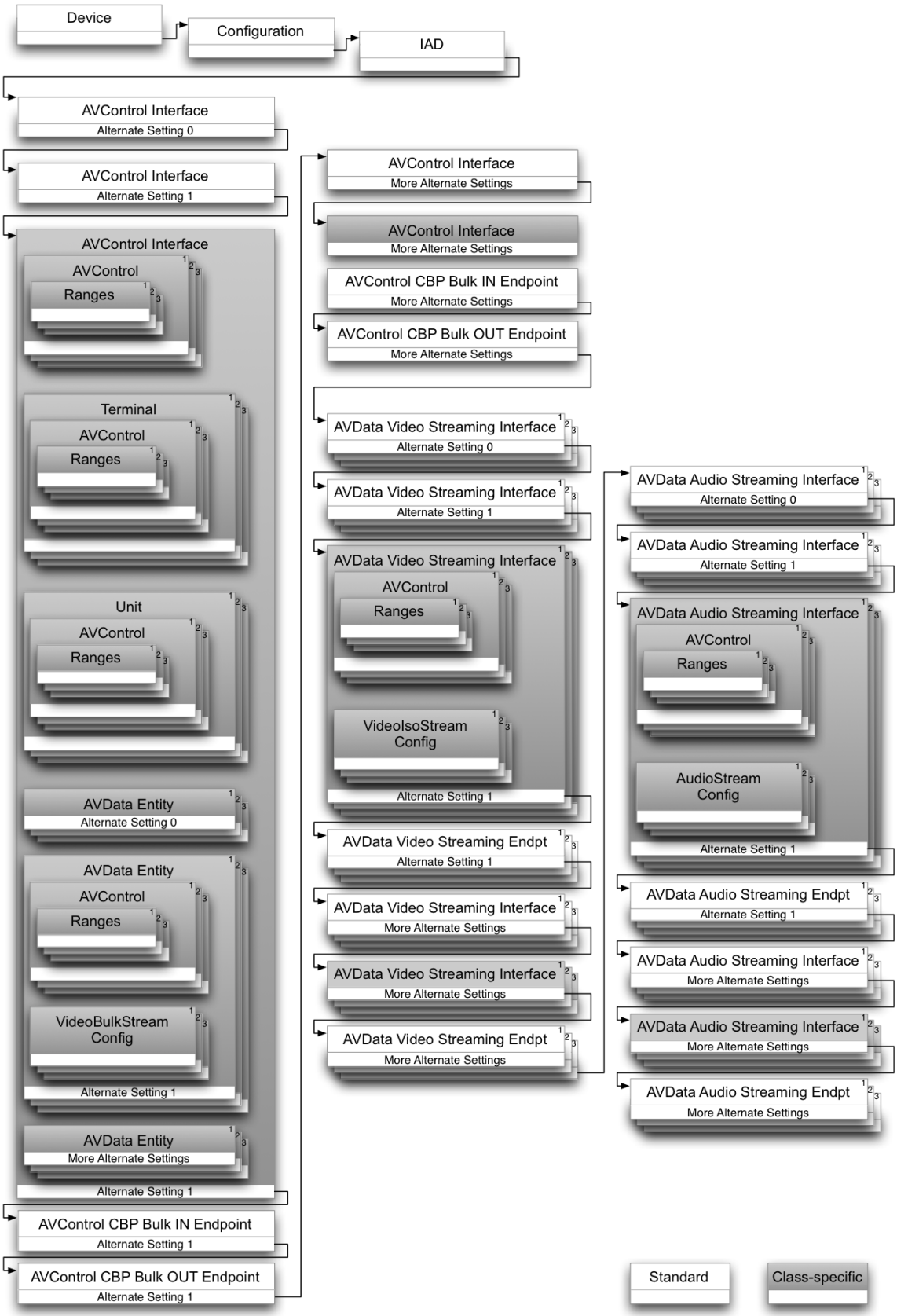


Figure 8-2: Example Legacy View Descriptor Hierarchy

8.4. Legacy View Descriptor Definitions

8.4.1. AVControl Interface Descriptor

The AVControl Interface Descriptor is the top-level Descriptor for all the class-specific aspects of the AVFunction. It includes information that applies globally to the AVFunction. This Descriptor is returned after the standard USB Interface Descriptor that corresponds to the AVControl Interface and is followed by a number of associated Descriptors that describe further detailed aspects of the AVControl Interface.

Table 8-1: AVControl Interface Descriptor

Offset	Field	Size	Value	Description
0	**bLength**	1	Number	Size of this Descriptor in bytes: 4.
1	**bDescriptorType**	1	Constant	AVCONTROL_IF Descriptor Type. See Appendix A.13.1, “Descriptor Type Codes”.
2	**bBCDAVFunction**	2	BCD	AV Class Definition Release Number in Binary-Coded Decimal (e.g., version 1.20 is 0x0120). This field identifies the release of the AV Class Definition with which the AVFunction and its Descriptors are compliant.

8.4.2. Terminal Descriptor

The Terminal Descriptor describes a Terminal (Input or Output Terminal). Whenever necessary, the Terminal Descriptor is followed by one or more AVControl Descriptors describing the AVControls supported by this Terminal.

Table 8-2: Terminal Descriptor

Offset	Field	Size	Value	Description
0	**bLength**	1	Number	Size of this Descriptor in bytes: 10.
1	**bDescriptorType**	1	Constant	TERMINAL Descriptor Type. See Appendix A.13.1, “Descriptor Type Codes”.
2	**wTerminalType**	2	Constant	Terminal Type: INPUT or OUTPUT. This field indicates whether this is an Input or Output Terminal. See Appendix A.13.2, “Terminal Type Codes” for allowed values.
4	**wEntityID**	2	Number	EntityID of this Terminal.
6	**wAssocEntityID**	2	Number	EntityID of the associated AVData Entity.
8	**wSourceID**	2	Number	EntityID of the source Entity to which the Input Pin of this Terminal is connected. Only applicable to Output Terminals. Reserved and shall be set to zero for Input Terminals.

8.4.3. Unit Descriptor

The Unit Descriptor describes a Unit. Whenever necessary, the Unit Descriptor is followed by one or more AVControl Descriptors describing the AVControls supported by this Unit.

Note that Legacy View restricts the number of advertised Input Pins to two (although more may be present – only exposed through the AVDD).

Table 8-3: Unit Descriptor

Offset	Field	Size	Value	Description
0	**bLength**	1	Number	Size of this Descriptor in bytes: 10.
1	**bDescriptorType**	1	Constant	UNIT Descriptor Type. See Appendix A.13.1, “Descriptor Type Codes”.
2	**wUnitType**	2	Constant	This field indicates the type of this Unit. See Appendix A.13.3, “Unit Type Codes” for allowed values.
4	**wEntityID**	2	Number	EntityID of this Unit.
6	**wSourceID1**	2	Number	EntityID of the source Entity to which Input Pin 1 of this Unit is connected.
8	**wSourceID2**	2	Number	EntityID of the source Entity to which Input Pin 2 of this Unit is connected. If the Unit has only one Input Pin, then the value of **bSourceID2** shall be set to 0x00.

8.4.4. AVControl Descriptor

The AVControl Descriptor describes an AVControl. Whenever necessary, the AVControl Descriptor is followed by one or more Ranges Descriptors describing the ranges supported by this AVControl.

Table 8-4: AVControl Descriptor

Offset	Field	Size	Value	Description
0	**bLength**	1	Number	Size of this Descriptor in bytes: 10.
1	**bDescriptorType**	1	Constant	AVCONTROL Descriptor Type. See Appendix A.13.1, “Descriptor Type Codes”.
2	**wControlSelector**	2	Constant	This field indicates the AVControl Type. Refer to Appendix A.11, “AVControl Selector Codes” for allowed values, depending on the Entity to which the AVControl belongs.
4	**wRWN**	1	Bitmap	This field indicates the Read/Write privileges for the CUR Property of the AVControl. It also indicates whether the NEXT Property is implemented. D1..0: 0b00: R – Read-Only 0b01: RW – Read-Write 0b10: W – Write-Only 0b11: Reserved. D2: 0b0: NEXT Property not implemented 0b1: NEXT Property implemented D15..3: Reserved. Shall be set to zero.

8.4.4.1. Ranges Descriptor

The Ranges Descriptor describes a range of values or a list of values that an AVControl supports. There are two variations of the Ranges Descriptor:

- RANGE Ranges Descriptor
- VALUELIST Ranges Descriptor

See Appendix A.13.5, “Ranges Codes” for the RANGE and VALUELIST definitions.

One or more of these Descriptors may be returned after the AVControl Descriptor of the AVControl that requires its ranges to be expressed. The rules for expressing ranges are exactly the same as those used in AVControl Descriptions. See Section 9.4, “Control Properties” for details.

When the **wRangeType** field is set to RANGE, then the Ranges Descriptor shall be in the format shown in Table 8-5. The RANGE Ranges Descriptor describes one range for the AVControl.

Table 8-5: RANGE Ranges Descriptor

Offset	Field	Size	Value	Description
0	**bLength**	1	Number	Size of this Descriptor in bytes: 16.
1	**bDescriptorType**	1	Constant	RANGE Descriptor Type. See Appendix A.13.1, “Descriptor Type Codes”.
2	**wRangeType**	2	Constant	Type of Range: RANGE. See Appendix A.13.5, “Ranges Codes”.
4	**dwMin**	4	Number	Minimum value for the range.
8	**dwMax**	4	Number	Maximum value for the range.
12	**dwResolution**	4	Number	Resolution of the range.

When the **wRangeType** field is set to VALUELIST, then the Ranges Descriptor shall be in the format shown in Table 8-6. The number of entries in a ValueList cannot exceed 62 ($n \leq 62$). If a ValueList needs more than 62 entries, it shall be broken down into smaller ValueLists made of no more than 62 entries, each requiring a separate VALUELIST Ranges Descriptor.

Table 8-6: VALUELIST Ranges Descriptor

Offset	Field	Size	Value	Description
0	**bLength**	1	Number	Size of this Descriptor in bytes: 4n+4.
1	**bDescriptorType**	1	Constant	RANGE Descriptor Type. See Appendix A.13.1, “Descriptor Type Codes”.
2	**wRangeType**	2	Constant	Type of Range: VALUELIST. See Appendix A.13.5, “Ranges Codes”.
4	**dwValue(1)**	4	Number	First value.
…	**..**	…	…	…
4n	**dwValue(n)**	4	Number	Last value.

8.4.5. AVData Entity Descriptor

The AVData Entity Descriptor describes an AVData Entity. The AVData Entity may represent a Source or Sink for AV content.

Whenever necessary, the AVData Entity Descriptor is followed by one or more AVControl Descriptors describing the AVControls supported by this AVData Entity.

If the AVData Entity is an AVData FrameBuffer entity and therefore able to interact with VideoBulkStreams, then the AVData Entity Descriptor is followed by one or more VideoBulkStreamConfig Descriptors.

If the AVData Entity is an AVData Video Streaming Interface and therefore able to interact with VideoIsoStreams, then the AVData Entity Descriptor is followed by one or more VideoIsoStreamConfig Descriptors.

If the AVData Entity is an AVData Audio Streaming Interface and therefore able to interact with AudioStreams, then the AVData Entity Descriptor is followed by one or more AudioStreamConfig Descriptors.

In Legacy View, an AVData Entity that is not an AVData Streaming Interface shall only support the (mandatory) Alternate Setting 0 and one Active Alternate Setting 1. An AVData Streaming Interface may support one or more Active Alternate Settings. However, since there is no class-specific information to convey for the Active Alternate Setting(s), there is no need for an Alternate Setting Descriptor.

Table 8-7: AVData Entity Descriptor

Offset	Field	Size	Value	Description
0	**bLength**	1	Number	Size of this Descriptor in bytes: 10.
1	**bDescriptorType**	1	Constant	AVDATA Descriptor Type. See Appendix A.13.1, “Descriptor Type Codes”.
2	**wEntityType**	2	Constant	This field indicates the Entity Type of this AVData Entity. See A.13.4, “AVData Entity Codes” for allowed values.
4	**wEntityID**	2	Number	EntityID of this Entity.
6	**wClockDomain**	2	Number	This field indicates how this AVData Entity is affiliated with a Clock Domain. 0: No Clock Domain affiliation. 1: Clock Source. 2: Clock Sink. All other values are reserved.
8	**wClockDomainID**	2	Number	This field indicates a unique ID for the Clock Domain to which this AVData Entity is affiliated.

8.4.6. VideoBulkStreamConfig Descriptor

The VideoBulkStreamConfig Descriptor describes one VideoBulkStream Configuration. For detailed information on VideoBulkStream Configurations, refer to [AVFORMAT_1].

Table 8-8: VideoBulkStreamConfig Descriptor

Offset	Field	Size	Value	Description
0	**bLength**	1	Number	Size of this Descriptor in bytes: 20.
1	**bDescriptorType**	1	Constant	VIDEOBULK Descriptor Type. See Appendix A.13.1, “Descriptor Type Codes”.
2	**bmAttributes**	2	Bitmap	This field indicates some characteristics of the VideoBulkStream Configuration as follows: D0: Reserved. Shall be set to zero. D1: HDCP: HDCP Protocol supported (D1=0b1) or not (D1=0b0). D15..2: Reserved. Shall be set to zero.
4	**wChannels**	2	Number	This field indicates the number of supported video channels in the single VideoTrack. Can be either 1 (2D – OL001) or 2 (3D2 – [OL001, OL002]).
6	**wVideoFrameRate**	2	Number	This field indicates the VideoFrame Rate of this VideoBulkStream Configuration. Shall be expressed in Hz.
8	**wFrameOrganization**	2	Number	This field indicates the Frame Organization of this VideoBulkStream Configuration. Shall be set to one of the allowed Frame Organization Legacy View Code (FOLVC) values as defined in [AVFORMAT_1].
10	**wFrameFormat**	2	Number	This field indicates the VideoFrame Format of this VideoBulkStream Configuration. Shall be set to one of the allowed VideoFrame Format Legacy View Code (VFLVC) values as defined in [AVFORMAT_1].
12	**wVideoSampleFormat**	2	Number	This field indicates the VideoSample Format of this VideoBulkStream Configuration. Shall be set to one of the allowed VideoSample Format Legacy View Code (VSLVC) values as defined in [AVFORMAT_1].
14	**wSubSlotSize**	2	Number	This field contains the SubSlot Size. Only applicable to Type I VideoSample Formats. Shall be set to 0 for all Type II VideoSample Formats (SubSlot Size is determined by specification).
16	**wBitResolution**	2	Number	This field contains the bit resolution of the VideoSample. Only applicable to Type I VideoSample Formats. Shall be set to 0 for all Type II VideoSample Formats.
18	**wVideoCompression**	2	Number	This field indicates the VideoCompression Method of this VideoBulkStream Configuration. Allowed values depend on the value of the **bVideoSampleFormat** field. Shall be set to one of the allowed Video Compression Legacy View Code (VCLVC) values as defined in [AVFORMAT_1].

8.4.7. VideoIsoStreamConfig Descriptor

The VideoIsoStreamConfig Descriptor describes one VideoIsoStream Configuration. For detailed information on VideoIsoStream Configurations, refer to [AVFORMAT_3].

Table 8-9: VideoIsoStream Config Descriptor

Offset	Field	Size	Value	Description
0	**bLength**	1	Number	Size of this Descriptor in bytes: 20.
1	**bDescriptorType**	1	Constant	VIDEOISO Descriptor Type. See Appendix A.13.1, “Descriptor Type Codes”.
2	**bmAttributes**	2	Bitmap	This field indicates some characteristics of the VideoIsoStream Configuration as follows: D0: Reserved. Shall be set to zero. D1: HDCP: HDCP Protocol supported (D1=0b1) or not (D1=0b0). D15..2: Reserved. Shall be set to zero.
4	**wChannels**	2	Number	This field indicates the number of supported video channels in the single VideoTrack. Can be either 1 (2D – OL001) or 2 (3D2 – [OL001, OL002]).
6	**wVideoFrameRate**	2	Number	This field indicates the VideoFrame Rate of this VideoIsoStream Configuration. Shall be expressed in Hz.
8	**wFrameOrganization**	2	Number	This field indicates the Frame Organization of this VideoIsoStream Configuration. Shall be set to one of the allowed Frame Organization Legacy View Code (FOLVC) values as defined in [AVFORMAT_3].
10	**wFrameFormat**	2	Number	This field indicates the VideoFrame Format of this VideoIsoStream Configuration. Shall be set to one of the allowed VideoFrame Format Legacy View Code (VFLVC) values as defined in [AVFORMAT_3].
12	**wVideoSampleFormat**	2	Number	This field indicates the VideoSample Format of this VideoIsoStream Configuration. Shall be set to one of the allowed VideoSample Format Legacy View Code (VSLVC) values as defined in [AVFORMAT_3].
14	**wSubSlotSize**	2	Number	This field contains the SubSlot Size.
16	**wBitResolution**	2	Number	This field contains the bit resolution of the VideoParticle.
18	**wVideoSIPSize**	2	Number	This field indicates the length in bytes of the SIP used to transport the VideoPayload.

8.4.8. AudioStreamConfig Descriptor

The AudioStreamConfig Descriptor describes one AudioStream Configuration. For detailed information on AudioStream Configurations, refer to [AVFORMAT_2].

Table 8-10: AudioStreamConfig Descriptor

Offset	Field	Size	Value	Description
0	**bLength**	1	Number	Size of this Descriptor in bytes: 20.
1	**bDescriptorType**	1	Constant	AUDIOISO Descriptor Type. See Appendix A.13.1, “Descriptor Type Codes”.
2	**bmAttributes**	2	Bitmap	This field indicates some characteristics of the AudioStream Configuration as follows: D0: Simple (D0=0b0) or Extended (D0=0b1) Audio Format. D1: HDCP: HDCP Protocol supported (D1=0b1) or not (D1=0b0). Only valid when D0=0b1. Shall be set to 0b0 when D0=0b0. D2: Timestamp: Timestamp Protocol supported (D2=0b1) or not (D2=0b0). Only valid when D0=0b1. Shall be set to 0b0 when D0=0b0. D15..3: Reserved. Shall be set to zero.
4	**wChannels**	2	Number	This field indicates the number of supported AudioChannels in the single AudioTrack. Can be either 2 (FL, FR) or 6 (FL, FR, FC, SL, SR, LFE).
6	**dAudioFrameRateMin**	4	Number	These fields indicate the minimum and maximum AudioFrame Rate of this AudioStream Configuration. Only useful when the Data endpoint supports a range of AudioFrame Rates (adaptive). In case of a non-adaptive endpoint, these fields shall be set to the same value. Shall be expressed in Hz.
10	**dAudioFrameRateMax**	4	Number	
14	**wAudioSampleFormat**	2	Number	This field indicates the AudioSample Format of this AudioStream Configuration. Shall be set to one of the allowed AudioSample Format Legacy View Code (AFLVC) values as defined in [AVFORMAT_2].
16	**wSubSlotSize**	2	Number	This field contains the SubSlot Size. Only applicable to Type I AudioSample Formats. Shall be set to 4 for all Type III AudioSample Formats.
18	**wBitResolution**	2	Number	This field contains the bit resolution of the AudioSample. Only applicable to Type I AudioSample Formats. Shall be set to 32 for all Type III AudioSample Formats.

9. Requests and Control Sequences

9.1. Standard Requests

The AV Device Class shall support at least the following standard requests described in Section 9, "USB Device Framework" of the USB Specification. The AV Device Class places no specific requirements on the values for the standard requests:

- Clear Feature
- Get Configuration
- Get Descriptor
- Get Interface
- Get Status
- Set Address
- Set Configuration
- Set Feature
- Set Interface

9.2. Class-Specific AVControl Sequences

Class-specific AV Control Sequences are used to set and get Property values of AVControls. All interaction with the internals of the AVFunction is performed via the manipulation of certain Properties of individual AVControls that are embedded in the Entities of the AVFunction. The Properties of an AVControl represents the finest level of interaction granularity available to the Controller. The class-specific AVFunction AV Description Document (AVDD) contains a collection of Entity descriptions, each indicating which AVControls are present in each Entity.

Control Sequences are *always* directed to the single AVControl Interface of the AV Function, irrespective of where the targeted AVControl resides. The Control Sequence contains enough information (EntityID, Control Selector, Property, Channel Number, etc.) for the AVFunction to decide where a specific Control Sequence shall be routed.

Note: Each addressable Entity in the AVFunction (Terminals, Units, AVData Entities, and Interfaces) shall have a class-specific assigned ID. EntityIDs shall be unique throughout the entire AVFunction. No two addressable Entities shall have the same EntityID, even if they cannot be in existence at the same time. EntityID zero is reserved and shall never be used for an addressable Entity inside the AVFunction. The mandatory AVControl Interface shall always have its EntityID set to one (0x0001) so that Controller software can access the AVControls inside the AVControl Interface Entity at all times.

Most AVControls and therefore their associated Control Sequences are optional. However, an AVFunction shall implement the AVControls and respond to the associated Control Sequences that are needed to operate that AVFunction. If an AVFunction does not support a certain Control Sequence, it shall indicate this by returning an Error Response Message. (See Section 9.3.2, "Response Message" for more details.)

9.3. AVControl Sequence Structure

The AVControl Sequence is used to set or get a Property of an AVControl inside an Entity of the AVFunction. This specification defines five types of Control Sequences. They all share a similar layout.

- Get Control Sequence (GET): This Sequence is used by the Controller to retrieve the value of a Property of an AVControl from the Controllee.
- Set with Return Control Sequence (SETR): This Sequence is used by the Controller to set the value of a Property of an AVControl in the Controllee and, in the same atomic operation, retrieve the value of that Property, potentially after it has been rounded and/or adjusted by the AVFunction in the Controllee to match the supported resolution and minimum and maximum values for that AVControl's Property.
- Set without Return Control Sequence (SET): This Sequence is used by the Controller to set the value of a Property of an AVControl in the Controllee without retrieving the rounded or adjusted value. This is useful for those cases where the value of the Property may consist of a rather large data set (such as the partial update of a SourceData Control) or where the value of the Property is used and discarded and not available for retrieval.

- Notify Control Sequence (NOTIF): This Sequence is used by the Controllee to notify the Controller that a Property value of an AVControl has changed through external means. The new value for the Property is included in the Notify Message.
- Null Message (NULL): This Sequence is used to indicate the end of a series of Messages or to poll the CBP.

If a certain SETR Control Sequence is supported, the associated SET and GET Control Sequence shall also be supported.

If a certain SET Control Sequence is supported, the associated GET and SETR Control Sequence may optionally be supported as a pair; i.e. if the GET Control Sequence is supported, then the SETR shall also be supported.

A certain GET Control Sequence may be supported without the associated SET and SETR Control Sequences being supported.

If certain AVControls can change via external means (for example, via a front panel button on the Device), then the appropriate NOTIF Control Sequence for these AVControls shall be supported.

A Control Sequence typically consists of 2 phases:

- In the first phase, the Controller sends a Command Message to the Controllee. The Command Message contains all the necessary addressing information for the Controllee to determine which Property of a particular AVControl is targeted and also, in case of the SET or SETR Control Sequence, the new desired value(s) for the AVControl's targeted Property.
- In a second phase, after decoding the received Command Message, the Controllee executes the command and returns an appropriate Response Message, either indicating a successful completion of the command or it detects an error and returns an Error Response Message. In the case of a GET or SETR Control Sequence, the Response Message also contains the value of the AVControl's targeted Property.

The Notify Control Sequence is a special case in that it consists of one phase only. In fact, it is a Message, sent from the Controllee to the Controller that is not triggered by a Command Message but rather by an event in the Controllee that caused a Property of a certain AVControl to change value. An AVControl shall generate a Notify Message whenever one or more of its Properties change value and the change is not a direct result of a Control Sequence issued to that AVControl. This can be either because an external event caused the Property value to change or because the AVFunction was forced to change the Property value as an indirect result of another Control Sequence. AVControls that can only change their Property values through direct Control Sequence manipulation shall never generate Notify Control Sequences.

In addition to the Command, Response, and Notify Message, this specification also defines a Null Control Sequence, which is also a one phase only Message and is used to indicate the end of a series of Control Sequences in some cases (see Section 7.3.1, "CBP Bulk Transfers" for details on the use of the Null Message).

Two-phase Control Sequences shall at all times remain in order. Response Messages shall be sent from Controllee to Controller in exactly the same order as their associated Command Messages were sent from Controller to Controllee. In other words, out-of-order processing of Control Sequences is not allowed.

All Messages consist of an invariant (INVAR) part followed by a variant (VAR) part.

All Messages shall be aligned on AV_MESSAGE_GRANULARITY-byte boundaries (see Appendix A.12, "AV Device Class General Constants"). The lNVAR part is always AV_MESSAGE_INVAR_SIZE bytes in length. Therefore, the VAR part shall always be (AV_MESSAGE_GRANULARITY- AV_MESSAGE_INVAR_SIZE) bytes plus an integer number (including zero) of AV_MESSAGE_GRANULARITY-byte segments. Depending on the actual parameter block length, the parameter block shall be padded with at most AV_MESSAGE_GRANULARITY-1 bytes to satisfy the AV_MESSAGE_GRANULARITY-byte aligment rule. It is not allowed to pad the parameter block with additional AV_MESSAGE_GRANULARITY-byte blocks.

The number of padding bytes (#PADDINGBYTES) and the length of the VAR part (VAR_LENGTH) can be calculated from the Data Length field (DATALEN) as follows:

```
TOTAL_LENGTH =     AV_MESSAGE_INVAR_SIZE + DATALEN;
PADDED_LENGTH =    ((TOTAL_LENGTH + AV_MESSAGE_GRANULARITY - 1) DIV
                   (AV_MESSAGE_GRANULARITY)) * AV_MESSAGE_GRANULARITY;
PADDING_BYTES =    PADDED_LENGTH - TOTAL_LENGTH;
```

9.3.1. Command Message

A Command Message is always sent from the Controller to the Controllee and consists of an invariant (INVAR) part followed by a variant (VAR) part. The invariant part layout is identical for all Command Messages whereas the variant part layout depends on the actual AVControl that is addressed. The following figure provides a detailed view on the Command Message layout. (In the figure, the length of the Control-specific part of the Message is chosen arbitrarily.)

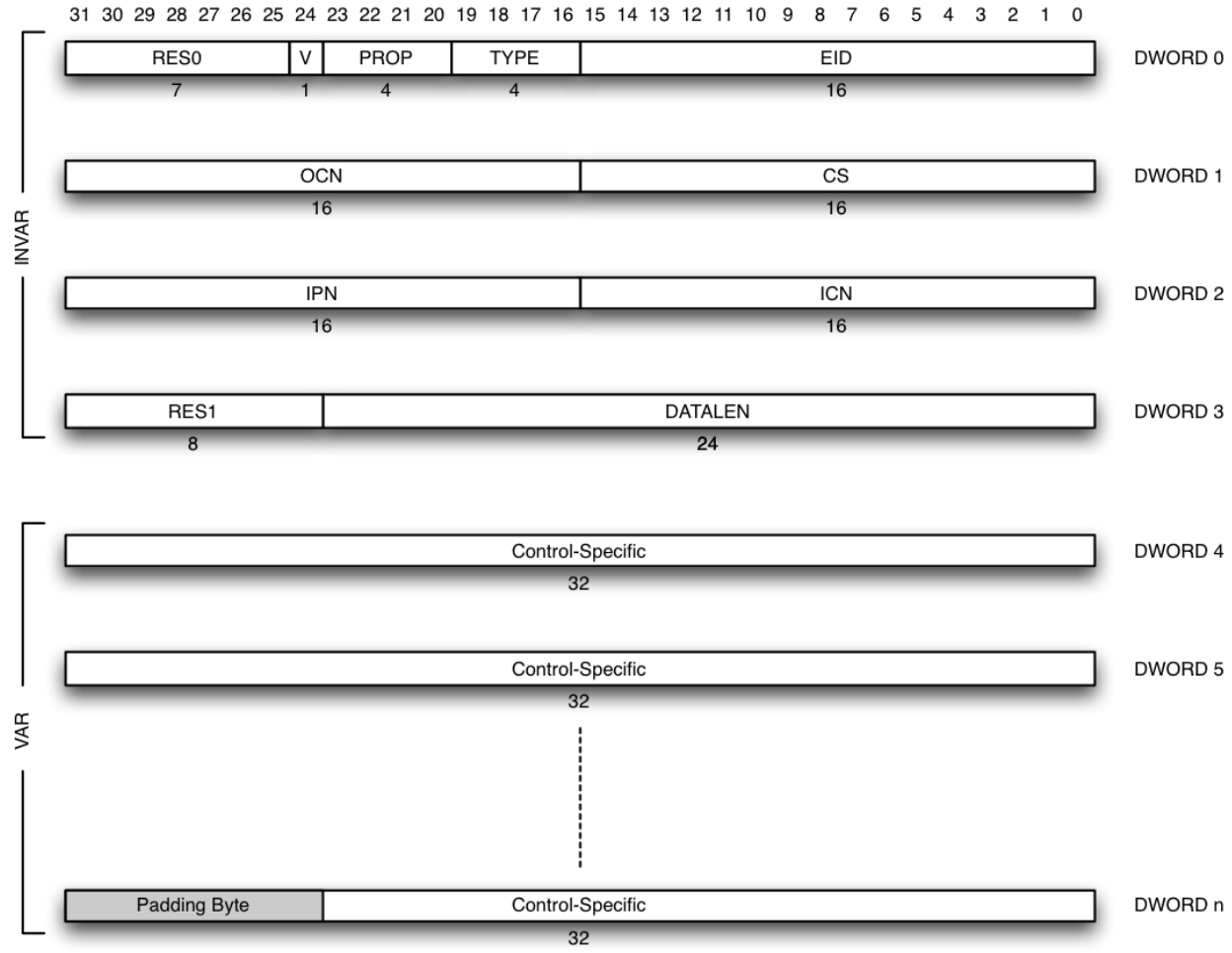


Figure 9-1: Command Message Layout

Note: The following paragraphs assume that the Vendor field (see below) is set to class-defined. For further information on vendor-defined Command Message layout, see Section 9.3.6, “Vendor-defined Messages”.

The INVAR part of the Command Message is always AV_MESSAGE_INVAR_SIZE bytes in length and contains the fields as described in the following paragraphs.

9.3.1.1. Message Fields

The EntityID (**EID**) field contains the unique ID of the Entity in which the AVControl resides. Specifying an unknown EntityID shall result in an Error Response Message.

The Type (**TYPE**) field indicates the type of Control Sequence that is being performed: SETR, SET, or GET. Specifying a value other than SETR, SET, or GET shall result in an Error Response Message. See Appendix A.9, "AV Request Codes" for possible values.

The Property (**PROP**) field indicates which Property of the AVControl is being addressed: CUR or NEXT. Specifying a value other than CUR or NEXT shall result in an Error Response Message. See Section 9.4, "Control Properties" for more details.

The Vendor (**V**) field indicates whether this is a class-specific Message (**V**=0) or a vendor-specific Message (**V**=1). See Section 9.3.6, "Vendor-defined Messages" for more details.

The Reserved0 (**RES0**) field is reserved for future extensions and shall be set to zero. Specifying a value other than zero shall result in an Error Response Message.

The Control Selector (**CS**) field indicates which type of AVControl within the Entity is adressed. Specifying a Control Selector value that is unknown to or not implemented by the Entity shall result in an Error Response Message. See Appendix A.11, "AVControl Selector Codes" for possible values.

The Input Pin Number (**IPN**) field contains the Input Pin Number of the input on which the AVControl resides. (In the case where the Entity that contains the AVControl only has one Input Pin, then the IPN field shall also be set to one – the Input Pin Number of the single Input Pin). Not all AVControls are associated with an Input Pin. If the AVControl is not associated with a particular Input Pin, then the IPN field shall be set to zero. Consequently, since no Input Pin association implies that no Channel association exists, the ICN and OCN fields shall also be set to zero. Specifying a value that is unknown to the Entity shall result in an Error Response Message.

The Input Channel Number (**ICN**) field contains the Input Channel Number of the Input Channel on which the AVControl resides (including Master Channel 0). Not all AVControls are associated with an Input Channel. If the AVControl is not associated with a particular Input Channel, then the ICN field shall be set to zero. Consequently, since no Input Channel association implies that no Output Channel association exists, the OCN field shall also be set to zero. Specifying a value that is unknown to the Entity shall result in an Error Response Message.

The Output Channel Number (**OCN**) field contains the Output Channel Number of the Output Channel on which the AVControl resides (including Master Channel 0). Not all AVControls are associated with an Output Channel. If the AVControl is not associated with a particular Output Channel, then the OCN field shall be set to zero. Specifying a value that is unknown to the Entity shall result in an Error Response Message.

The sections that describe all AVControls in detail indicate for each AVControl which of the IPN-ICN-OCN fields is relevant for that AVControl. Unused IPN-ICN-OCN fields shall be set to zero by the sender of the Message. The receiver of the Message shall ignore the values in the unused IPN-ICN-OCN fields.

The VAR part contains the actual parameter block of the Message. The type of the AVControl determines the length and layout of the parameter block. This layout is specified in Sections 9.7, "Common AVControls" and following where each AVControl is described in detail. The Data Length (**DATALEN**) field indicates the length of the parameter block, expressed in bytes.

The Reserved1 (**RES1**) field is reserved for future extensions and shall be set to zero. Specifying a value other than zero shall result in an Error Response Message.

9.3.2. Response Message

A Response Message is always sent from the Controllee to the Controller in response of a Command Message. Its structure is very similar to the Command Message and it also consists of an invariant (INVAR) part followed by a variant (VAR) part. The invariant part layout is identical for all Response Messages whereas the variant part layout depends on the actual AVControl that is addressed. The following figure provides a detailed view of the Response Message layout. (In the figure, the length of the Control-specific part of the Message is chosen arbitrarily.)

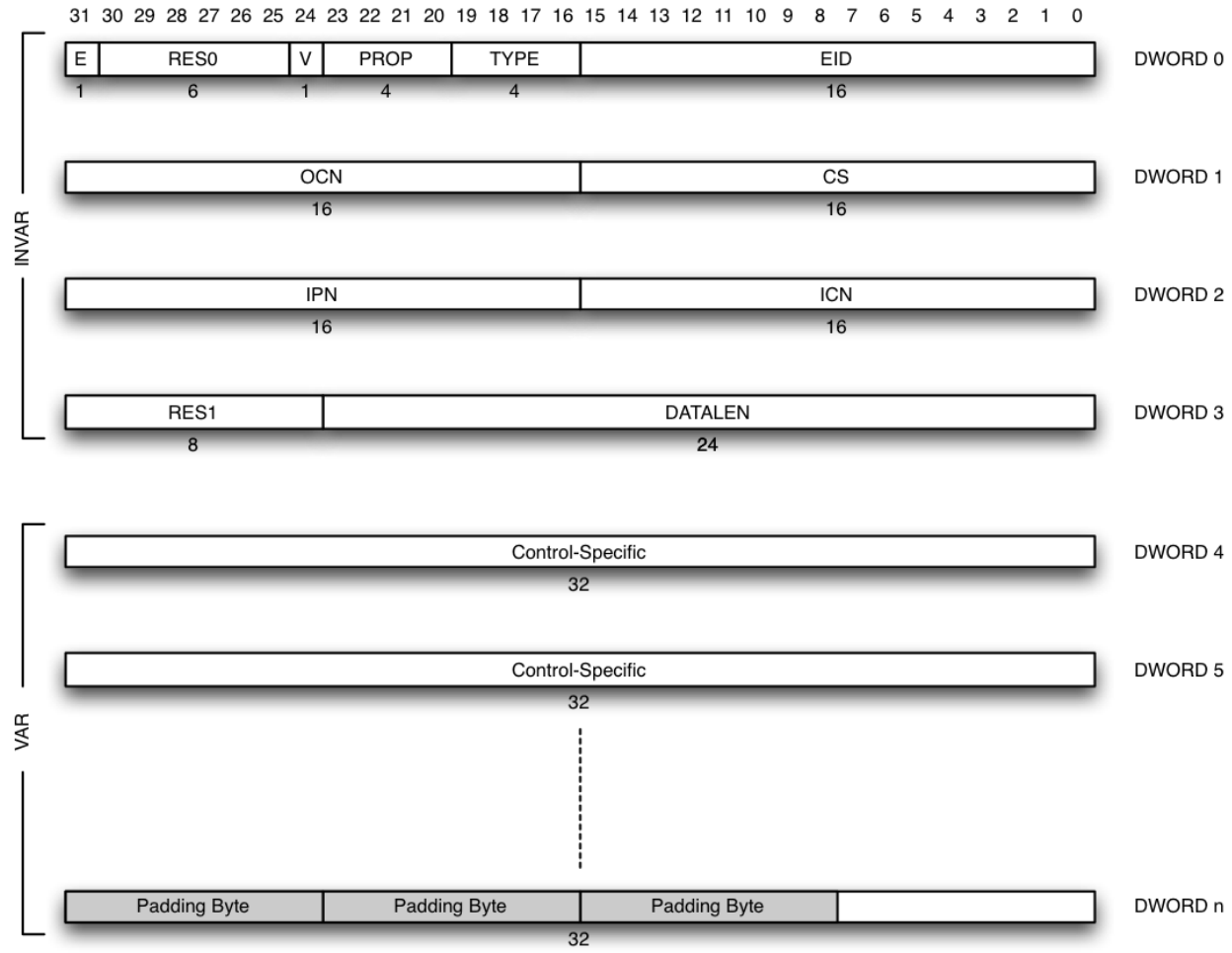


Figure 9-2: Response Message Layout

Note: The following paragraphs assume that the Vendor field is set to class-defined. For further information on vendor-defined Command Message layout, see Section 9.3.6, “Vendor-defined Messages”.

The INVAR part of the Response Message is always AV_MESSAGE_INVAR_SIZE bytes in length and contains the fields as described in the following paragraphs.

For details about Response Message fields not explained below, see Section 9.3.1.1, “Message Fields”.

The Response Message is generated after the Controllee has executed the command it received in a previous Command Message. It therefore mainly serves as an acknowledge message from the Controllee to the Controller. The INVAR part of the Response Message is almost identical to the INVAR part of the associated Command Message. All fields in the INVAR part of the Response Message, besides the Error Field (non-existent in the Command Message) and potentially the Data Length field, shall contain the same values as the corresponding fields in the associated Command Message. The values in these fields can be used as a signature to associate Command and Response Messages at the Controller side.

The INVAR part of the Response Message contains an Error (**E**) field.

Whenever the Controllee detects an error condition, it shall return a Response Message as described above but with the Error field set to one. Also, in this case, the Response Message shall not contain a VAR part and therefore the Data Length field shall be set to zero as well. This type of Response Message is called the Error Response Message.

If the Command was executed successfully, the Controllee shall return a Response Message as described above but now with the Error field set to zero.

For GET and SETR Control Sequences, the VAR part contains the actual parameter block of the Response Message. The type of the AVControl determines the length and layout of the parameter block. This layout is specified in Sections 9.7, “Common AVControls” and following where each AVControl is described in detail. The Data Length (**DATALEN**) field indicates the length of the parameter block, expressed in bytes.

For SET Control Sequences, the Response Message does not contain a VAR part and the Data Length field of the Response Message shall be set to zero.

9.3.3. Notify Message

A Notify Message is always sent from the Controllee to the Controller and consists of an invariant (INVAR) part followed by a variant (VAR) part. The invariant part layout is identical for all Notify Messages whereas the variant part layout depends on the actual AVControl that is addressed. The following figure provides a detailed view on the Notify Message layout. (In the figure, the length of the Control-specific part of the Message is chosen arbitrarily.)

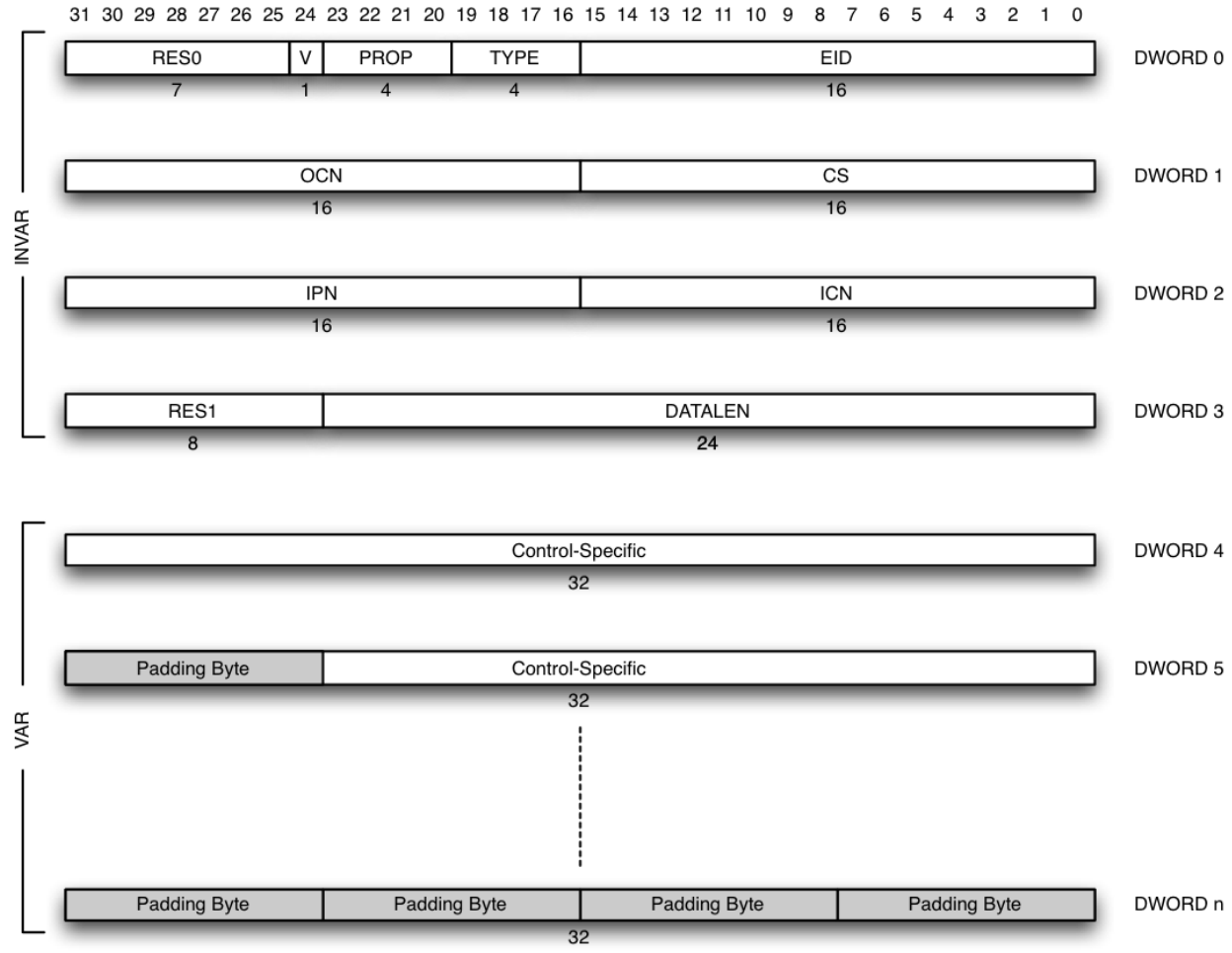


Figure 9-3: Notify Message Layout

Note: The following paragraphs assume that the Vendor field is set to class-defined. For further information on vendor-defined Command Message layout, see Section 9.3.6, “Vendor-defined Messages”.

Note that the layout of the Notify Message is identical to the layout of the Command Message. The only difference is that the Command Message is sent from the Controller to the Controllee and the Notify Message is sent in the opposite direction.

The INVAR part of the Notify Message is always AV_MESSAGE_INVAR_SIZE bytes in length and contains the fields as described in the following paragraphs.

For details about Notify Message fields not explained below, see Section 9.3.1.1, “Message Fields”.

The Type (**TYPE**) field indicates the type of Control Sequence that is being performed and shall be set to NOTIF. See Appendix A.9, “AV Request Codes” for the NOTIF value.

9.3.4. Null Message

A Null Message can be sent from the Controller to the Controllee or from the Controllee to the Controller and consists of an invariant (INVAR) part only. The Null Message indicates the end of a series of Control Sequences or is used as a keep-alive message and therefore does not carry any further meaningful information (see Section 7.3.1, “CBP Bulk Transfers” for details on the use of the Null Message).

The following figure provides a detailed view on the Null Message layout.

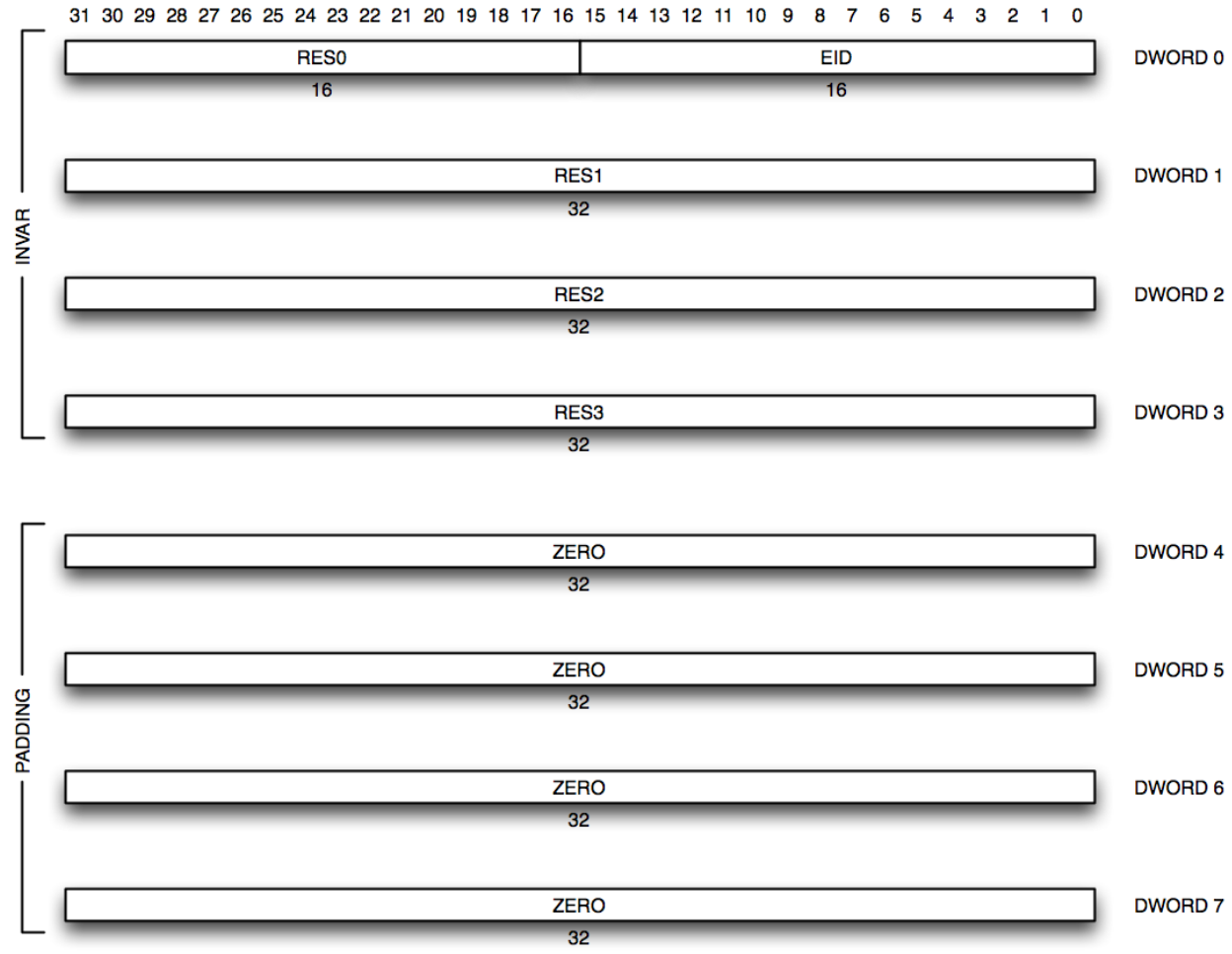


Figure 9-4: Null Message Layout

The INVAR part of the Null Message is always AV_MESSAGE_INVAR_SIZE bytes in length and contains the fields as described in the following paragraphs.

The EntityID (**EID**) field contains the value zero (0x0000) which uniquely identifies this Message as the Null Message.

The Reserved0 (**RES0**) and Reserved1 (**RES1**) through Reserved3 (**RES3**) fields are reserved for future extensions and shall be set to zero.

The Null Message is further padded with AV_MESSAGE_GRANULARITY – AV_MESSAGE_INVAR_SIZE bytes to satisfy the AV_MESSAGE_GRANULARITY-byte alignment requirement.

9.3.5. Control Sequence Examples

The following sections graphically illustrate the different types of Control Sequences.

9.3.5.1. Set with Return (SETR) Control Sequence

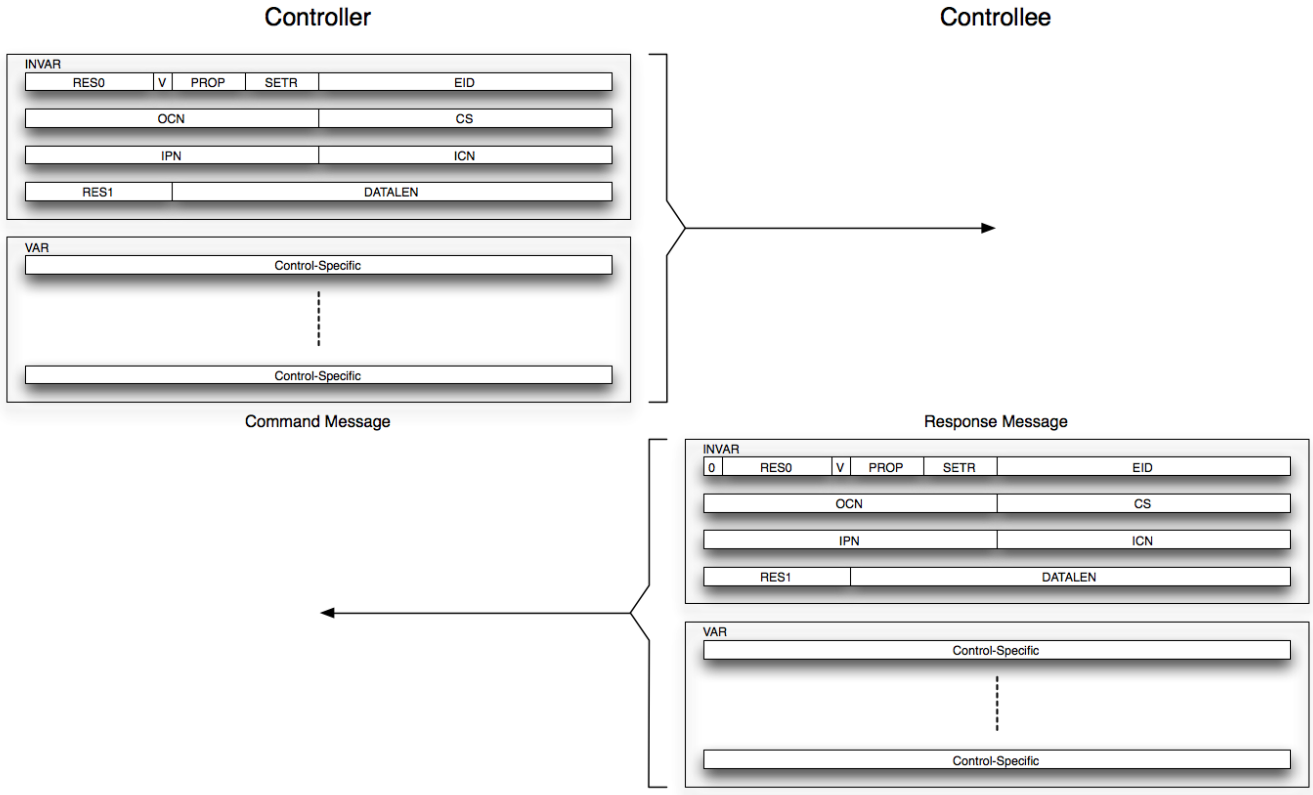


Figure 9-5: SETR Control Sequence

9.3.5.2. Set with Return (SETR) Control Sequence with Error

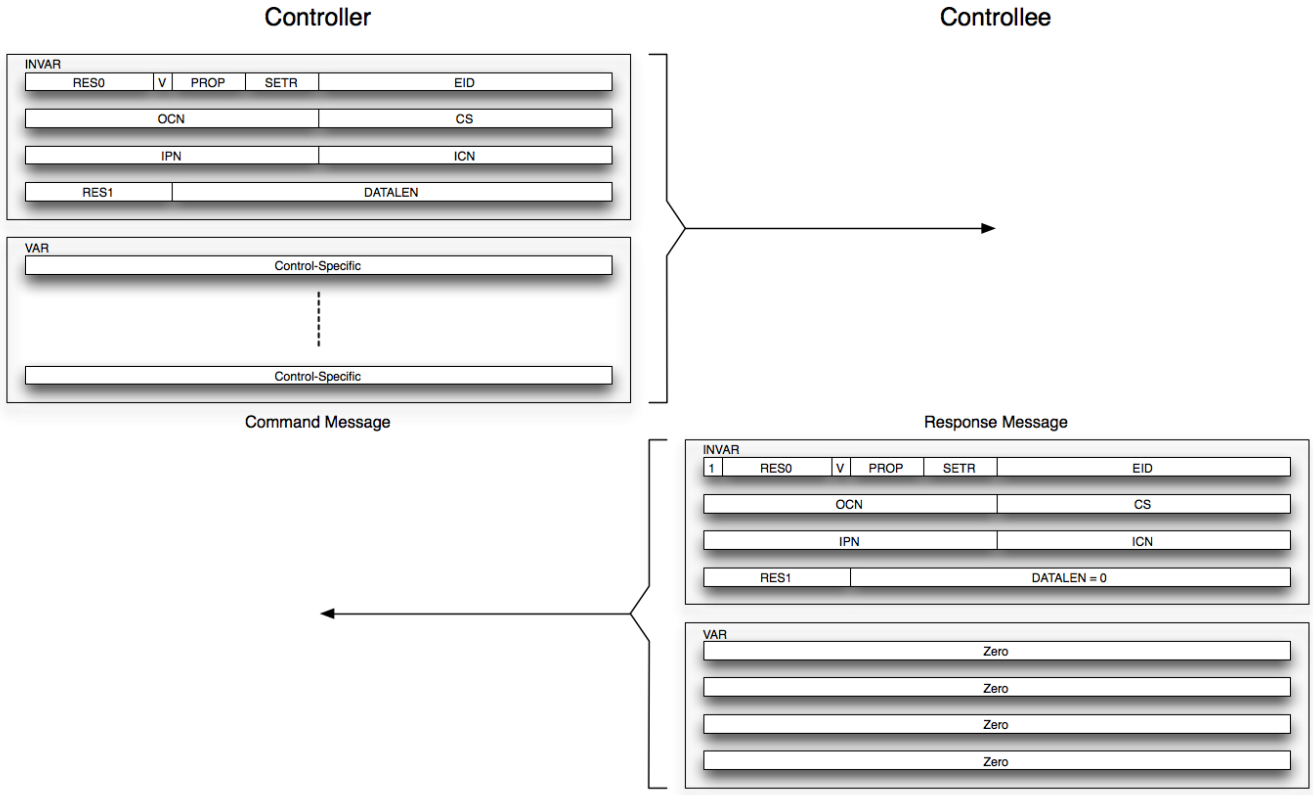


Figure 9-6: SETR Control Sequence with Error

9.3.5.3. Set without Return (SET) Control Sequence

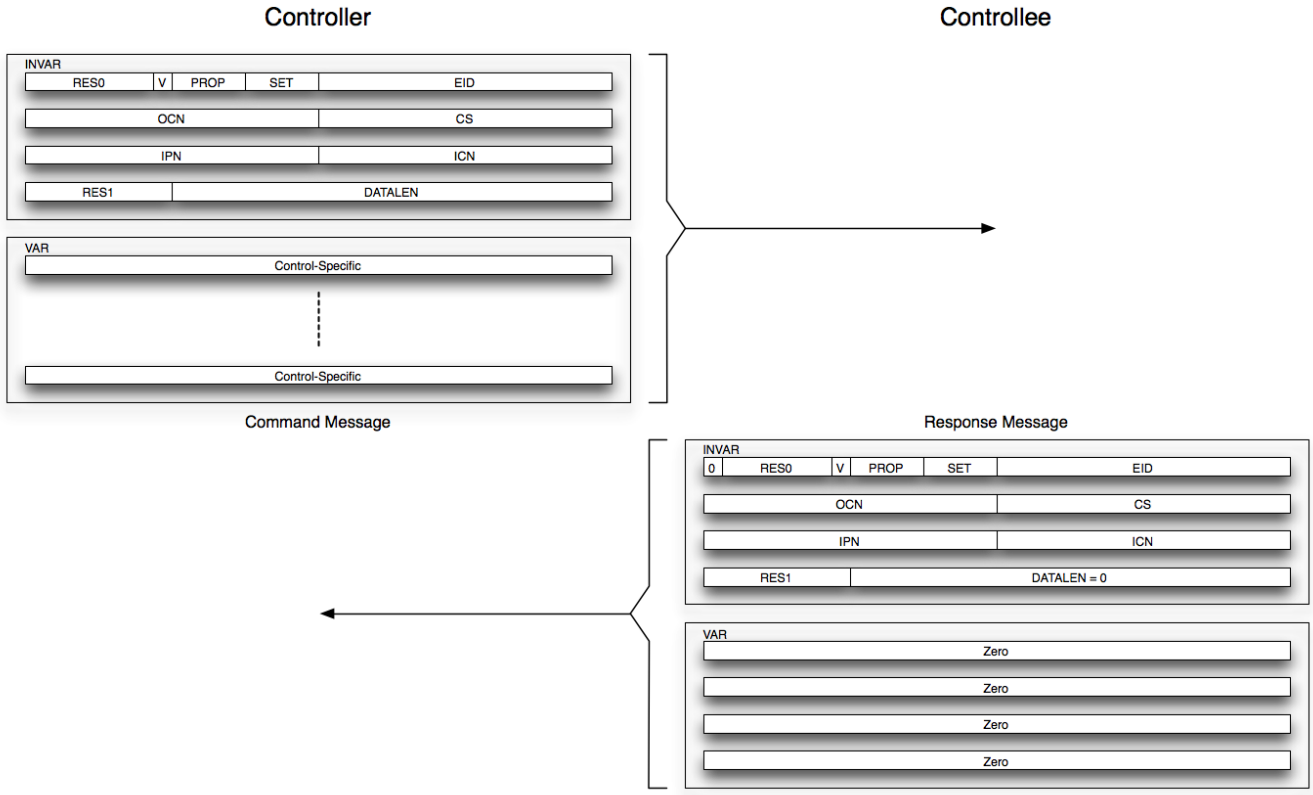


Figure 9-7: SET Control Sequence

9.3.5.4. Set without Return (SET) Control Sequence with Error

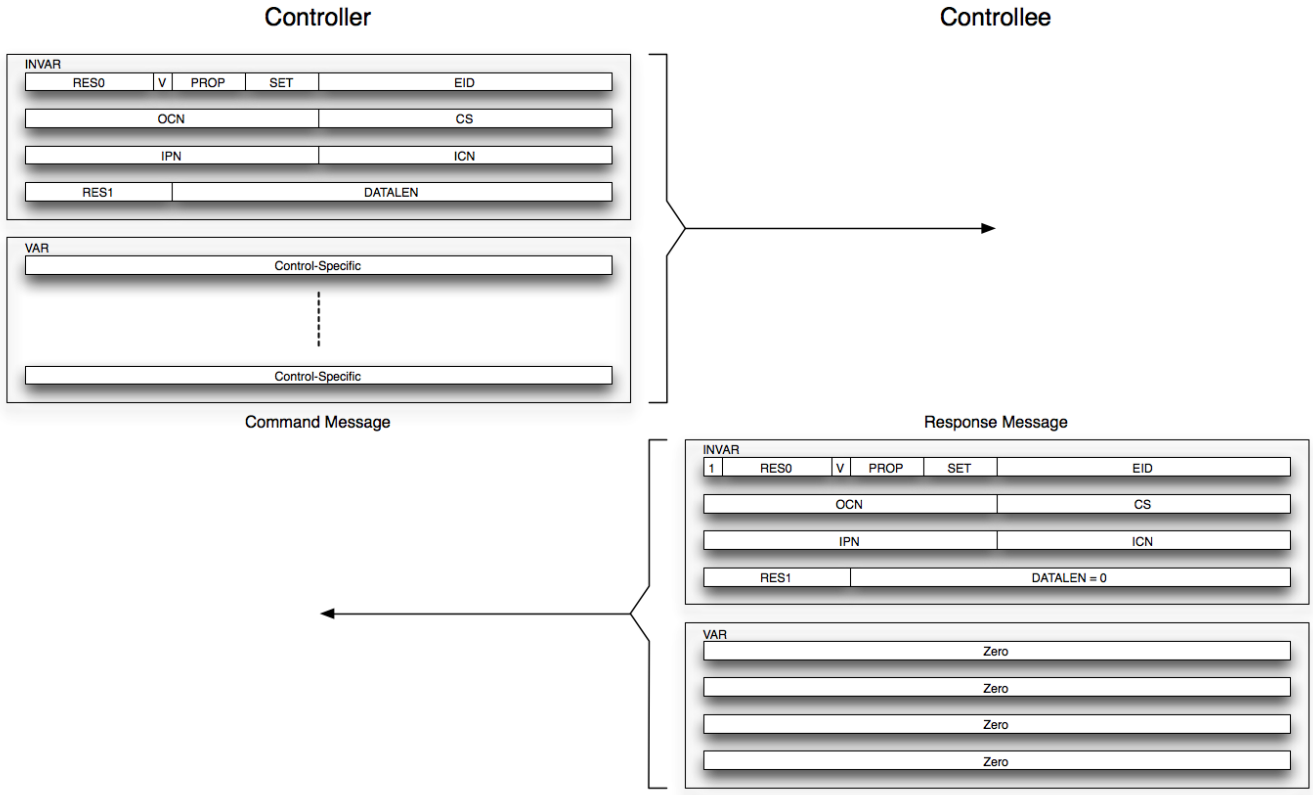


Figure 9-8: SET Control Sequence with Error

9.3.5.5. Get (GET) Control Sequence

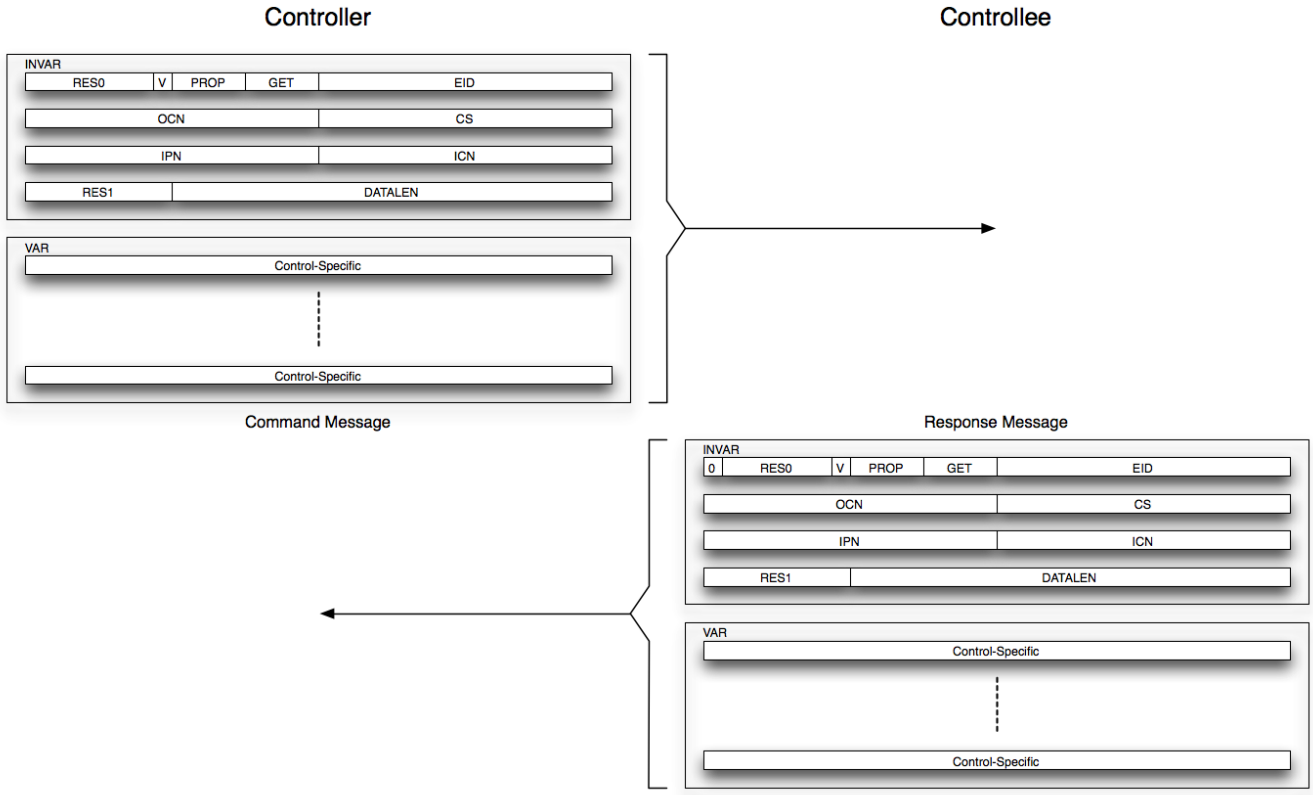


Figure 9-9: GET Control Sequence

9.3.5.6. Get (GET) Control Sequence with Error

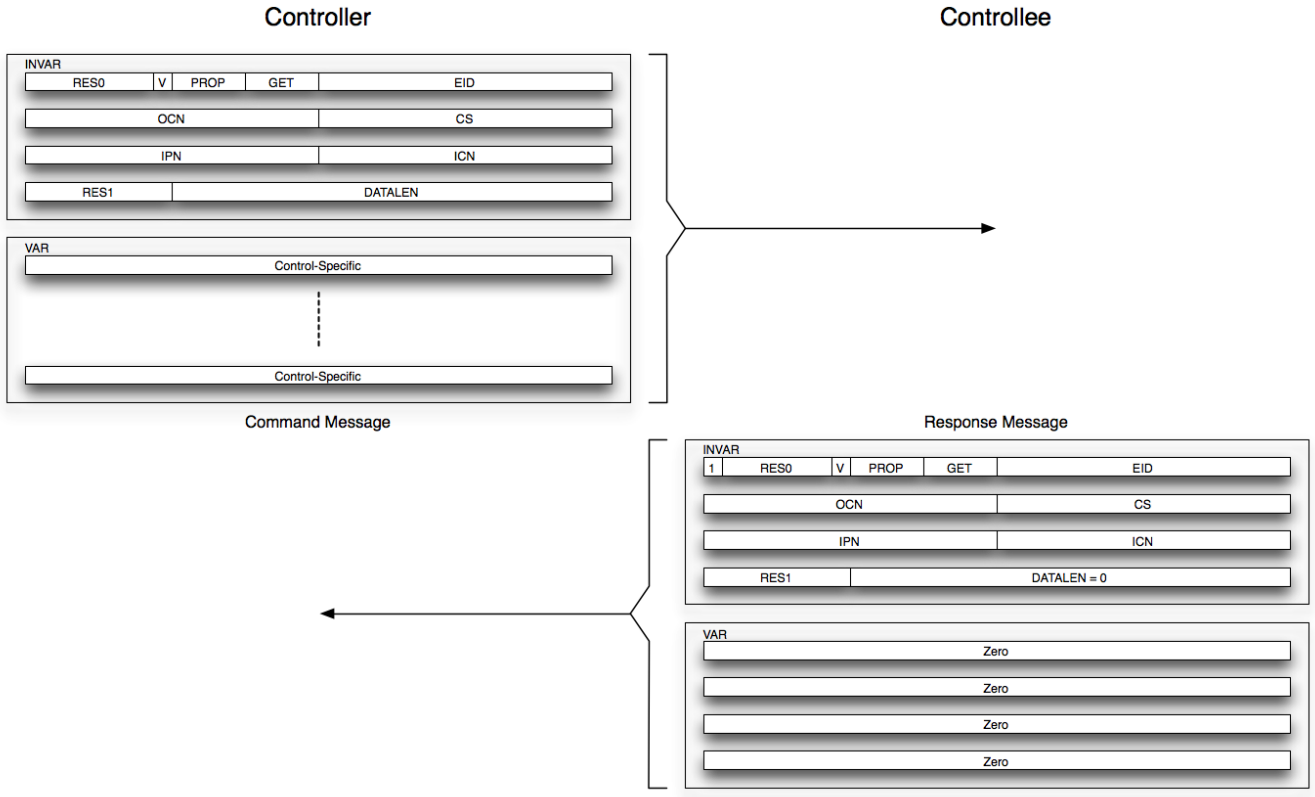


Figure 9-10: GET Control Sequence with Error

9.3.5.7. Notify Control Sequence

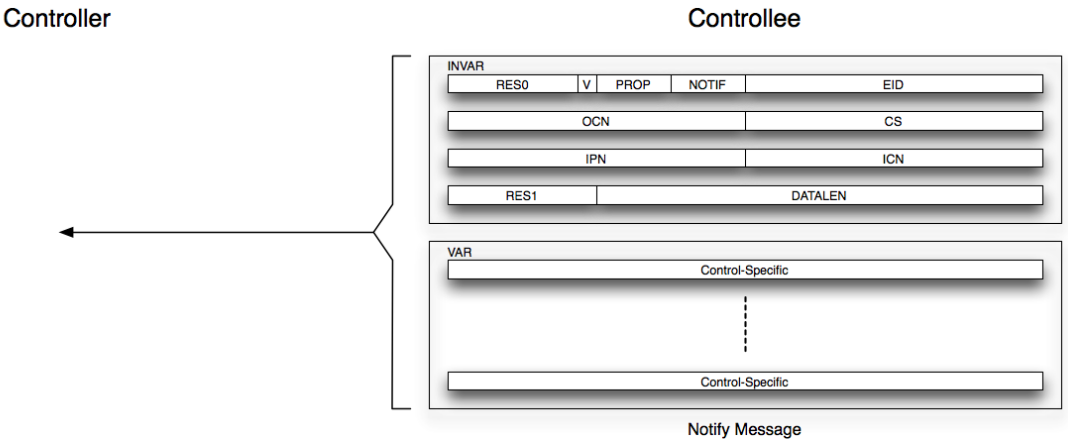


Figure 9-11: NOTIF Control Sequence

9.3.6. Vendor-defined Messages

When the Vendor (**V**) field in a Message is set to one (**V**=1), the Message is vendor-defined. However, it is still required that vendor-defined Messages follow the same general structural layout as their class-defined counterparts. In other words, all the INVAR part fields (EID, TYPE, PROP, DATALEN, etc.) shall still be present. However, the allowed values in those fields may deviate from the class-defined allowed values. Furthermore, the CS, IPN, ICN, and OCN fields may be repurposed as necessary although it is strongly recommended that the AVControl paradigm be maintained in the vendor-defined space and therefore, the structures to address those vendor-defined AVControls (CS, IPN, ICN, OCN)

should be maintained as well. The rules associated with Control Sequences (Control-Response Message binding, in-order messaging, etc.) shall also apply form vendor-defined Control Sequences.

9.4. Control Properties

Each AVControl within an Entity can have one or more Properties associated with it. Currently defined Properties for an AVControl are its:

- Current setting Property (CUR)
- Next setting Property (NEXT)
- Ranges Property (<ranges>)

The CUR and NEXT Properties are manipulated by issuing the appropriate Control Sequence to the targeted AVControl. The <ranges> Property is static and is reported as part of the class-specific AV Description Document (AVDD) of the AVFunction.

This specification defines the Read/Write privileges of an AVControl's Properties as follows:

- This specification defines some AVControls as Read-Only (R). For this type of AVControls, the CUR Property shall be Read-Only and the NEXT Property shall not be supported.
- This specification defines most AVControls as Read-optional-Write (RoW). This means that a particular implementation can decide whether to implement the AVControl's CUR Property either as Read-Write (RW) or as Read-Only (R). Most AVControls that implement their CUR Property as Read-Only usually do not support the NEXT Property. However, some AVControls may implement their CUR Property as Read-Only (R) and also provide a NEXT Property (always implemented as Read-Write (RW)) and rely on the presence and use of the Commit AVControl to change value. In other words, these AVControls cannot change their CUR Property directly but only through an update via the NEXT Property followed by a Commit.
  For AVControls that are implemented as Read-Write (RW), the NEXT Property may be supported and shall always be implemented a Read-Write (RW).
- This specification defines a few AVControls as Write-optional-Read (WoR). This means that a particular implementation can decide whether to implement the AVControl's CUR Property either as Write-Only (W) or as Read-Write (RW). For this type of AVControl, the NEXT Property shall not be supported.
- The Commit AVControl is the only AVControl defined as Write-Only (W) by this specification. The Commit AVControl's CUR Property shall be implemented as Write-Only (W) and the NEXT Property shall not be supported.
- The <ranges> Property is Read-Only by construction.

Properties may be implemented as Read-Only (R) or Read-Write (RW). In addition, some Properties may be implemented as Write-Only (W). This applies to Properties that typically consist of a large set of values and once written, keeping the values around for reading purposes would impose unwarranted buffer requirements on the AVFunction without actual benefit. (A FrameBuffer SourceData Control is a typical example). However, implementations may choose to provide a readback option for these Properties anyway and declare them as Read-Write (RW).

Note: For Properties that are allowed by specification to be implemented as either Read-Only (R) or Read-Write (RW), it is important to make the distinction the Read/Write privilege the specification allows, expressed by the word Read-optional-Write (RoW) and the actual Read/Write privilege an AVFunction chooses for that Property, expressed by the words Read-Only (R) or Read-Write (RW).
Likewise, for Properties that are allowed by specification to be implemented as either Write-Only (W) or Read-Write (RW), it is important to make the distinction the Read/Write privilege the specification allows, expressed by the word Write-optional-Read (WoR) and the actual Read/Write privilege an AVFunction chooses for that Property, expressed by the words Write-Only (W) or Read-Write (RW).

In summary, the specification may allow the Read/Write privilege of a Property to be Read-Only (R), Read-optional-Write (RoW), Read-Write (RW), Write-Only (W), or Write-optional-Read (WoR). An implementation shall always express the Read/Write privilege of a Property as Read-Only (R), Read-Write (RW), or Write-Only (W).

The CUR Property is used to manipulate the current actual setting of an AVControl. Manipulating this Property has immediate effect on the actual setting of the AVControl. However, in many cases, it is desirable to apply settings to multiple AVControls simultaneously. For example, changing the settings for a Parametric Equalizer Section (Center Frequency, Q Factor, and Gain) sequentially rather than simultaneously may introduce totally undesired side effects and artifacts. To cover these cases, this specification supports the NEXT Property and a function-wide Commit Control Sequence. Manipulating the NEXT Property 'preprograms' the AVControl with a next value without changing the CUR Property. The actual setting of the underlying (hardware) control is also not influenced. Only after the Commit Control Sequence completes successfully will the NEXT Properties be applied to the CUR Properties for all the AVControls in the AV Function, effectively changing all the AVControls in the AVFunction simultaneously (within the limits of the underlying firmware and hardware). The NEXT and CUR Properties of an AVControl will now have the same value. Note that manipulating the CUR Property of an AVControl also has an effect on its NEXT Property. Whenever the value of the CUR Property is changed, the NEXT Property takes on that same value (i.e. the NEXT and CUR Property are kept in sync when the CUR Property is manipulated). This guarantees that subsequent Commit Control Sequences will only have an effect on those AVControls that had their NEXT Property changed since the previous Commit Request was issued. However, this also implies that the NEXT Property cannot be used to hold a 'default' or 'preset' value that can be activated by a Commit Control Sequence. AVControls that always *have* to be manipulated simultaneously can declare their CUR Properties as Read-Only and solely rely on the update mechanism described above (preprogramming all the NEXT Properties involved, followed by a Commit Control Sequence).

The <ranges> Property provides information about the limitations the AVControl imposes on the allowed settings of the CUR and NEXT Properties.

The <ranges> Property is reported as a child element of the AVControl's XML representation in the AV Description Document, and shall contain zero or more <range> child elements and zero or one <valueList> child element. At least one <range> or one <valueList> child element shall always be present.

The <range> element in turn shall contain one or more occurrences of the <min>, <max>, and <res> triplet child elements. The <min>, <max>, and <res> triplet defines a subrange for the AVControl's allowed values. Multiple occurrences of the <range> element allow for accurate reporting of discontinuous multiple subranges for an AVControl's allowed values. <range> elements (subrange triplets) shall be ordered in ascending order (from lower values to higher values). Individual subranges shall not overlap (i.e. the <max> element value of the previous <range> element cannot be greater than or equal to the <min> element value of the next <range> element). The actual valid values defined by a <range> element (subrange triplet) are defined by the following pseudo-code:

```
i = 0; ValidArray(i) = <min>; NextValid = <min> + <res>;
while (NextValid <= <max>)
   {i++; validArray(i) = NextValid; NextValid = NextValid + <res>}
```

The <valueList> element contains a list of discrete values (one or more) that are allowed for the AVControl's value. The <range> and <valueList> element may both be present. However, values in the <valueList> shall never coincide with an allowed value as indicated by the <range> elements.

The general form of the <ranges> element looks as follows:

```
<ranges>
   <range>
      <min>xxx</min>
      <max>yyy</max>
      <res>zzz</res>
   </range>
   <!-- zero or more occurrences of the range element -->
            ...
   <valueList>uuu vvv ... </valueList>
   <!-- zero or one occurrence of the valueList element -->
<ranges>
```

Example:

Consider a (hypothetical) Volume Control that can take the following values for its CUR Property:

- -100 dB to -80 dB in steps of 10 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 `<ranges>` Property is then:

```
...
<ranges>
   <range>
      <min>-100</min>
      <max>-80</max>
      <res>10</res>
   </range>
   <range>
      <min>-70</min>
      <max>-40</max>
      <res>3</res>
   </range>
   <range>
      <min>-38</min>
      <max>-20</max>
      <res>2</res>
   </range>
   <range>
      <min>-19</min>
      <max>0</max>
      <res>1</res>
   </range>
</ranges>
...
```

Another way of representing the same Volume Control is as follows:

```
...
<ranges>
   <range>
      <min>-70</min>
      <max>-43</max>
      <res>3</res>
   </range>
   <range>
      <min>-40</min>
      <max>-22</max>
      <res>2</res>
   </range>
   <range>
      <min>-20</min>
      <max>0</max>
      <res>1</res>
   </range>
   <valueList>-100 -90 -80</valueList>
</ranges>
...
```

Note: The above examples are for illustrating purposes only. The actual values should be expressed as 16.16 Signed Fixed Point values. For example, -100 dB should be represented as -100*65536 = -6553600 (= 0xFF9C0000) but for readability, the value -100 is used.

It is left to the designer to choose a suitable representation. While it is possible to represent a range as the discrete values that make up that range, this representation is highly discouraged, especially when the number of discrete values is large.

In general, support for any of the possible Properties is optional. However, some AVControls require a minimum set of Properties in order to be functional. In the sections that follow and that describe the AVControls in details, mandatory support for certain Properties is explicitly called out.

9.4.1. Single Value Read-Only AVControl Considerations

In a particular AVFunction implementation, some Read-Only AVControls may only take a single value. This type of AVControl is conceptually present only to report its single value. Since that value is available through the `<ranges>` construct in the AVDD, there is no need for the AVFunction to actually support a GET Control Sequence (CUR) for that AVControl. An AVFunction implementation may therefore choose to respond with an Error Response Message if a Controller were to issue a GET Control Sequence (CUR) for that AVControl.

9.5. Parameter Block Layout

With a few exceptions, almost all Control Sequences involve a single AVControl Property during a SET/SETR, GET or NOTIF Control Sequence. For those Control Sequences, there is a single default Parameter Block (PB) layout defined to set or retrieve a value for the CUR or NEXT Property. The same default layout is used for the SET/SETR, GET, and NOTIF Control Sequences.

All Property values are expressed as 32-bit quantities. The actual value representation can take several forms, depending on the type of AVControl:

- Unsigned Integer: the decimal point is located to the right of the least significant bit and the potential value range of the Property is [0 (0x00000000) – 4,294,967,295 (0xFFFFFFFF)].
- Signed Integer: the decimal point is located to the right of the least significant bit and the potential value range of the Property is [-2,147,483,648 (0x80000000) – 2,147,483,647 (0x7FFFFFFF)].
- Unsigned Fixed Point (16.16): the decimal point is located between bit 15 and bit 16 and the potential value range of the Property is [0.00 (0x0000.0000) – 65,535.9999847 (0xFFFF.FFFF)]
- Signed Fixed Point (16.16): the decimal point is located between bit 15 and bit 16 and the potential value range of the Property is [-32768.00 (0x8000.0000) – 32767.9999847 (0x7FFF.FFFF)].
- 32-bit bitmap: The semantics of the bits are specific to a particular AVControl.

Important Note: All values reported through the AVDD shall be expressed in decimal format, either as Unsigned Integer (range [0 (0x00000000) – 4,294,967,295 (0xFFFFFFFF)]) or as Signed Integer (range [-2,147,483,648 (0x80000000) – 2,147,483,647 (0x7FFFFFFF)]), depending on whether the value is signed or unsigned. For values that are actually Unsigned Fixed Point (16.16) or Signed Fixed Point (16.16), the real value is obtained by dividing the reported decimal value by 65,536 (2^{16}).

For those Control Sequences that use a deviating PB layout, the actual layout is explicitly defined in the relevant sections and can potentially be different for the SET, SETR, GET and NOTIF Control Sequences. However, unless otherwise indicated, the same layout is used for the SET, SETR, GET and NOTIF Control Sequences for a particular AVControl.

The next paragraph specifies the default PB layout for the CUR or NEXT Property.

9.5.1. Default PB Layout

The default PB for the CUR or NEXT Property of an AVControl is always 4 bytes in size (DATALEN=4 in the appropriate Message) and has the following layout:

Table 9-1: CUR or NEXT PB Layout

Offset	Field	Size	Value	Description
0	dCUR or dNEXT	4	Number	The setting for the CUR or NEXT Property of the addressed AVControl.

9.6. AVControl Sections Layout

The layout of the following sections that describe the possible AVControls in more detail is as follows:

First, there is a paragraph briefly describing the functionality of the AVControl. Most AVControls are optional for implementations to support. However, some AVControls shall be implemented by all class-compliant AVFunction implementations. Those AVControls are clearly marked as mandatory at the beginning of this description paragraph.

Then a standardized table follows that summarizes a number of different aspects of the AVControl and applicable values for the different fields in the INVAR and PB of the Control Sequence that is used to manipulate the AVControl.

The first rows of the table enumerate the possible Properties (CUR, NEXT) and for each Property, the required support level is indicated in the L column. Mandatory (M) indicates that the AVControl shall support this Property. Optional (O) indicates that the AVControl may choose to support the Property, while Prohibited (P) indicates that the AVControl is not allowed to support the Property.

The RW column indicates the level of freedom an implementation has regarding the Read-Write privilege of the Property: whether the Property can be implemented as Read-Only (R), Read-Write (RW), Read with optional Write (RoW), Write-Only (W), or Write with optional Read (WoR):

- R means that the Property shall be implemented as Read-Only (no choice).
- RW means that the Property shall be implemented as Read-Write (no choice).
- RoW means that the implementation may choose to implement the Property as either Read-Only or Read-Write.
- W means that the Property shall be implemented as Write-Only (no choice).
- WoR means that the implementation may choose to implement the Property as either Write-Only or Read-Write.

The next 4 rows together specify the range of values that are applicable for the CUR and NEXT Property. As indicated above, a combination of a number of `<range>` elements and a `<valueList>` element describe the allowed values for the CUR and NEXT Properties. If zero or more `<range>` elements are allowed, this is indicated by the notation `<range>*`. If only the one `<range>` element is allowed, the asterisk (*) is omitted and the `<valueList>` row is not present. If zero or one `<valueList>` elements are allowed, this is indicated by the notation `<valueList>?`. If only the one `<valueList>` element is allowed, the question mark (?) is omitted and the `<range>`, `<min>`, `<max>`, and `<res>` rows are not present. For AVControls that have a range that is fixed by this specification, the `<range>`, `<min>`, `<max>`, `<res>`, and `<valueList>` rows are not present.

The CS row contains the Control Selector value that is to be used for that particular AVControl.

The IPN-ICN-OCN row contains information about the triplet address values that can be used to target a particular AVControl within the Entity.

Finally, if the AVControl has a deviating PB layout (other than the default PB layout), that layout is explicitly listed in subsequent rows, as applicable.

9.6.1. Example 1: AVControl Table that uses the default PB layout

Table 9-2: AVControl Table with Default Layout

Property	RW	L	
CUR	R/RW/RoW/W/WoR	M/O/P	
NEXT	RW	M/O/P	
<range>*	**<min>**		Applicable range
	<max>		Applicable range
	<res>		Applicable range
<valueList>?			Applicable range
Field			**Value**
CS			Control Selector Value
IPN-ICN-OCN			a-b-c

9.6.2. Example 2: AVControl Table with explicit PB layout

Table 9-3: AVControl Table with Explicit PB

Property	RW	L			
CUR	R/RW/RoW/W/WoR	M/O/P			
NEXT	RW	M/O/P			
<range>*	**<min>**		Applicable range		
	<max>		Applicable range		
	<res>		Applicable range		
<valueList>?			Applicable range		
Field	**Value**				
CS	Control Selector Value				
IPN-ICN-OCN	a-b-c				
PB	**DATALEN**	Length in bytes of the CUR or NEXT PB			
Offset	**Field**		**Size**	**Value**	**Description**
0	Property Name		x	As Applicable	…
…	…		…	…	…

9.7. Common AVControls

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

Note: The XX_ prefix that appears in a number of constants shall be replaced by the appropriate two-letter abbreviation for the particular Entity to which the Control Sequence was issued.

9.7.1. Cluster Control

The Cluster Control is used to indicate the configuration of the logical AVCluster that is currently present at the Output Pin of a particular Entity. The CUR Property returns an XML Document that complies with the syntax for an AVCluster as defined in [AVSCHEMA]–<avCluster>. Whenever the AVCluster configuration changes, the Cluster Control shall generate a Notify Message to inform the Controller of the change. The Cluster Control can also be queried via a GET Control Sequence at all times to retrieve the current AVCluster configuration.

Table 9-4: Cluster Control Table

Property	RW	L	Valid Range		
CUR	R	M	-		
NEXT	-	P	-		
Field			**Value**		
CS			XX_CLUSTER		
IPN-ICN-OCN			Note-0-0		
PB	**DATALEN**		n		
Offset	**Field**	**Size**	**Value**	**Description**	
0	**avClusterDescription**	n	String	The avCluster XML Document.	

Note: For the GraphicsEngine Entity, the IPN field shall contain the number of the **Output** Pin for which the AVCluster XML Document is to be returned. In all other cases, the IPN shall be set to zero.

The following is an example instance XML Document that complies with the AV Schema definition. It describes an AVCluster with 2 VideoTracks; the first VideoTrack contains 3D2 16:9 video content and the second VideoTrack contains 2D 4:3 video content. In addition, the AVCluster contains 2 AudioTracks; the first AudioTrack contains 5.1 audio content in US English and the second AudioTrack contains stereo (2.0) audio content in BE Dutch. Finally, the AVCluster contains 2 MetadataTracks; the first MetadataTrack contains one Subtitles MetadataChannel in US English and the second MetadataTrack also contains one Subtitles MetadataChannel in BE Dutch:

```
<?xml version="1.0" encoding="UTF-8"?>
<av:avCluster xmlns:av="http://www.usb.org/schemas/AVSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://www.usb.org/schemas/AVSchema
http://www.usb.org/schemas/AVSchema.xsd">
   <videoCluster>
      <videoTrack>
         <channels>
            <OL001/>
            <OL002/>
         </channels>
      </videoTrack>
      <videoTrack>
         <channels>
            <OL001/>
         </channels>
      </videoTrack>
   </videoCluster>
   <audioCluster>
      <audioTrack>
         <channels>
            <FL/>
            <FR/>
            <FC/>
            <SL/>
            <SR/>
            <LFE/>
         </channels>
         <languageID>en-US</languageID>
      </audioTrack>
      <audioTrack>
         <channels>
            <FL/>
            <FR/>
```

```
            </channels>
            <languageID>nl-BE</languageID>
         </audioTrack>
      </audioCluster>
      <metadataCluster>
         <metadataTrack>
            <channels>
               <SUB/>
            </channels>
            <languageID>en-US</languageID>
         </metadataTrack>
         <metadataTrack>
            <channels>
               <SUB/>
            </channels>
            <languageID>nl-BE</languageID>
         </metadataTrack>
      </metadataCluster>
</av:avCluster>
```

9.7.2. VideoTrack Selector Control

The VideoTrack Selector Control is used to indicate (Read-Only) the current or select (Read-Write) the desired VideoTrack from the available VideoTracks in the incoming VideoCluster. The number of VideoTracks from which to select can be retrieved from the AVCluster Description of the current incoming AVCluster. If the currently selected Active VideoTrack becomes obsolete due to a change in AVCluster configuration, the Controller will be informed of this change through a Cluster Control Notify Message and should take appropriate action (for example, selecting another Active VideoTrack).

Note: It is assumed that underlying hardware/software/firmware that implements the VideoTrack Selector Control is built such that it supports at least the maximum number of VideoTracks that can ever be present in any potential AVCluster that may be present on the associated Input Pin.

Table 9-5: VideoTrack Selector Control Table

Property	RW	L	Valid Range
CUR	RoW	M	-
NEXT	RW	O	-
<range>	**<min>**		1 (0x00000001)
	<max>		[1 (0x00000001) – 65,535 (0x0000FFFF)]
	<res>		1 (0x00000001)
Field			**Value**
CS			XX_VIDEO_TRACK_SELECTOR
IPN-ICN-OCN			Input Pin#-0-0

9.7.3. AudioTrack Selector Control

The AudioTrack Selector Control is used to indicate (Read-Only) the current or select (Read-Write) the desired AudioTrack from the available AudioTracks in the incoming AudioCluster. The number of AudioTracks from which to select can be retrieved from the AVCluster Description of the current incoming AVCluster. If the currently selected Active AudioTrack becomes obsolete due to a change in AVCluster configuration, the Controller will be informed of this change through a Cluster Control Notify Message and should take appropriate action (for example, selecting another Active AudioTrack).

Note: It is assumed that underlying hardware/software/firmware that implements the AudioTrack Selector Control is built such that it supports at least the maximum number of AudioTracks that can ever be present in any potential AVCluster that may be present on the associated Input Pin.

Table 9-6: AudioTrack Selector Control Table

Property	RW	L	Valid Range
CUR	RoW	M	-
NEXT	RW	O	-
<range>	**<min>**		1 (0x00000001)
	<max>		[1 (0x00000001) – 65,535 (0x0000FFFF)]
	<res>		1 (0x00000001)
Field			**Value**
CS			XX_AUDIO_TRACK_SELECTOR
IPN-ICN-OCN			Input Pin#-0-0

9.7.4. MetadataTrack Selector Control

The MetadataTrack Selector Control is used to indicate (Read-Only) the current or select (Read-Write) the desired MetadataTrack from the available MetadataTracks in the incoming MetadataCluster. The number of MetadataTracks from which to select can be retrieved from the AVCluster Description of the current incoming AVCluster. If the currently selected Active MetadataTrack becomes obsolete due to a change in AVCluster configuration, the Controller will be informed of this change through a Cluster Control Notify Message and should take appropriate action (for example, selecting another Active MetadataTrack).

Note: It is assumed that underlying hardware/software/firmware that implements the MetadataTrack Selector Control is built such that it supports at least the maximum number of Metadata Tracks that can ever be present in any potential AVCluster that may be present on the associated Input Pin.

Table 9-7: MetadataTrack Selector Control Table

CUR	RoW	M	-
NEXT	RW	O	-
<range>	<min>		1 (0x00000001)
	<max>		[1 (0x00000001) – 65,535 (0x0000FFFF)]
	<res>		1 (0x00000001)
CS			XX_METADATA_TRACK_SELECTOR
IPN-ICN-OCN			Input Pin#-0-0

9.8. AVControl Interface Controls

The following paragraphs present a detailed description of all possible AVControls an AVControl Interface can incorporate. For each AVControl, the supported Properties and their value ranges are specified. Also, the appropriate Control Selector value and IPN-ICN-OCN values are listed. The Control Selector codes are defined in Appendix A.11.1, "AVControl Interface Control Selectors".

9.8.1. AVDD Controls

All class-specific configuration information regarding the AVFunction is bundled within a single XML Document, the AV Description Document (AVDD). Retrieval of the document is accomplished by first issuing a GET Control Sequence to the AVDD Info Control. This AVControl returns the total length of the AVDD and also a flag indicating whether the AVDD has changed since the last time it was read. Reading the AVDD itself is accomplished by issuing a GET Control Sequence to the AVDD Content Control.

The following sections define the two AVControls that are required to retrieve the AVDD from the AVFunction. Since this is an essential capability of any AVFunction, all implementations shall support the AVDD Info Control and the AVDD Content Control.

9.8.1.1. AVDD Info Control

The AVDD Info Control returns the dAVDDLength field followed by the bmAVDDFlags field. The dAVDDLength field contains the total length of the AVDD, expressed in bytes. The bmAVDDFlags field indicates in D0 whether the AVDD has changed since the last time it was read (D0=0b1) or not (D0=0b0).

The AVDD Info Control shall also be used to generate a Notify Message whenever the AVFunction has a need to report a change in its class-specific configuration (to support dynamic configuration changes). In response to such a Notify Message, the Controller should first bring the AVFunction into an idle state before retrieving the updated AVDD via the AVDD Content Control. The AVDD Info Control can also be queried via a GET Control Sequence at all times to retrieve its state.

All AVFunction implementations shall support this AVControl.

Table 9-8: AVDD Info Control Table

Property	RW	L	Valid Range
CUR	R	M	-
NEXT	-	P	-

Field	Value
CS	AC_AVDD_INFO
IPN-ICN-OCN	0-0-0
PB	**DATALEN** 8

Offset	Field	Size	Value	Description
0	dAVDDLength	4	Number	The total length of the AVDD, expressed in bytes.
4	bmAVDDFlags	4	Bitmap	D0: When set, indicates that the AVDD has been updated since the last time it was read. D31..1: Reserved. Shall be set to zero.

9.8.1.2. AVDD Content Control

The AVDD Content Control returns the actual AV Description Document, expressed in XML. The Length of the AVDD can be determined by first reading the AVDD Info Control and retrieving the total length, n, from the dAVDDLength field.

The AVDD Content Control shall never be used to notify the Controller of a change in configuration. Instead, the AVDD Info Control shall be used for that purpose.

All AVFunction implementations shall support this AVControl.

Table 9-9: AVDD Content Control Table

Property	RW	L	Valid Range		
CUR	R	M	-		
NEXT	-	P	-		
Field			**Value**		
CS			AC_AVDD_CONTENT		
IPN-ICN-OCN			0-0-0		
PB	**DATALEN**		8		
Offset	**Field**		**Size**	**Value**	**Description**
0	AVDDContent		n	-	The actual content of the AVDD, expressed in XML.

9.8.2. Commit Control

The Commit Control is different from all other AVControls. It does not represent a physical AVControl in the AVFunction. Instead, it is used to simultaneously update the CUR Properties of all the AVControls within the entire AVFunction with their corresponding NEXT Property values in a synchronized fashion. The Commit Control can do this either asynchronously (CUR = 0), i.e. at the moment the Commit Control receives the Set Control Sequence, or synchronized to an event, such as the occurrence of the vertical sync interval, for example (CUR = 1). In that case, the Set Control Sequence acts as the enabler for the commit to take place at the next occurrence of the event.

The Commit Control is a Write-Only AVControl. Issuing a Get Control Sequence shall result in an Error Response Message.

Support for the Commit Control is conditionally required. If none of the AV Function's Controls support the NEXT Property, then this request shall not be supported. However, if one or more Controls do support the NEXT Property, then the Commit Control shall be supported.

Table 9-10: Commit Control Table

Property	RW	L	Valid Range
CUR	W	M	-
NEXT	-	P	-
<range>	**<min>**		0 (0x00000000)
	<max>		[0 (0x00000000) – 1 (0x00000001)]
	<res>		1 (0x00000001)
Field			**Value**
CS			AC_COMMIT
IPN-ICN-OCN			0-0-0

9.8.3. Power Line Frequency Control

The Power Line Frequency Control is used to inform the AVFunction what the power line frequency is of the electrical environment in which the AVFunction is operating.

Note: The AVFunction can use this information to adjust certain internal algorithms and processes, such as flicker reduction filters, etc. If the AVFunction is connected to mains power, then it could automatically detect the current frequency. In that case, the Power Line Frequency Control may be Read-Only or it can be omitted entirely.

Support for this AVControl is optional.

Table 9-11: Power Line Frequency Control Table

Property	RW	L	Valid Range
CUR	RoW	M	-
NEXT	RW	O	-
<range>*	**<min>**		[0.00 (0x0000.0000) – 65,535.9999847 (0xFFFF.FFFF)]
	<max>		[0.00 (0x0000.0000) – 65,535.9999847 (0xFFFF.FFFF)]
	<res>		[0.000015258789 (0x0000.0001) – 65,535.9999847 (0xFFFF.FFFF)]
<valueList>?			[0.00 (0x0000.0000) – 65,535.9999847 (0xFFFF.FFFF)]
Field			**Value**
CS			AC_POWER_LINE_FREQ
IPN-ICN-OCN			0-0-0

9.8.4. Remote Only Control

The Remote Only Control is used to disable all local controls (front panel buttons, etc.) on the AVFunction. When enabled, the Controller has exclusive control of all the AVControls inside the AVFunction. No interaction with the AVFunction other than through Control Sequences is allowed. (For example, all front panel buttons are disabled).

Support for this AVControl is optional.

Table 9-12: Remote Only Control Table

Property	RW	L	Valid Range
CUR	RW	M	FALSE/TRUE (0x00000000/0x00000001) – FALSE=Disabled; TRUE=Enabled.
NEXT	RW	O	FALSE/TRUE (0x00000000/0x00000001)
Field			**Value**
CS			AC_REMOTE_ONLY
IPN-ICN-OCN			0-0-0

9.9. Terminal Controls

The following paragraphs present a detailed description of all possible AVControls a Terminal can incorporate. For each AVControl, the supported Properties and their value ranges are specified. Also, the appropriate Control Selector value and the IPN-ICN-OCN values are listed. The Control Selector codes are defined in Appendix A.11.2, "Terminal Control Selectors"

9.9.1. Input Terminal

9.9.1.1. Cluster Control

All Input Terminals shall support this AVControl.

See Section 9.7.1, "Cluster Control" for a detailed description. "XX" in that description shall be replaced by "TE".

9.9.2. Output Terminal

There are no AVControls currently defined for Output Terminals.

9.10. Mixer Unit Controls

The following paragraphs present a detailed description of all possible AVControls a Mixer Unit can incorporate. For each AVControl, the supported Properties and their value ranges are specified. Also, the appropriate Control Selector value and the IPN-ICN-OCN values are listed. The Control Selector codes are defined in Appendix A.11.3, "Mixer Unit Control Selectors".

9.10.1. Video Controls

This version of the specification does not provide support for this item.

9.10.2. Audio Controls

9.10.2.1. AudioTrack Selector Control

See Section 9.7.3, "AudioTrack Selector Control" for a detailed description. "XX" in that description shall be replaced by "MU".

9.10.2.2. Level Control

The Level Control is used to control the sound level of a particular Input Channel that is mixed into a particular Output Channel.

All Audio Mixer Units shall support this AVControl.

Table 9-13: Level Control Table

Property	RW	L	Valid Range
CUR	RoW	M	-
NEXT	RW	O	-
<range>*	**<min>**		[-32,768.00 dB (0x8000.0000) – 32767.9999847 dB (0x7FFF.FFFF)]
	<max>		[-32,768.00 dB (0x8000.0000) – 32767.9999847 dB (0x7FFF.FFFF)]
	<res>		[0.000015258789062 dB (0x0000.0001) – 32767.9999847 dB (0x7FFF.FFFF)]
<valueList>?			[-32,768.00 dB (0x8000.0000) – 32767.9999847 dB (0x7FFF.FFFF)]
Field			**Value**
CS			MU_LEVEL
IPN-ICN-OCN			Input Pin#-Input Channel#-Output Channel#

9.11. Metadata Unit Controls

This version of the specification does not provide support for this item.

9.12. Selector Unit Controls

The following paragraphs present a detailed description of all possible AVControls a Selector Unit can incorporate. For each AVControl, the supported Properties and their value ranges are specified. Also, the appropriate Control Selector value and the IPN-ICN-OCN values are listed. The Control Selector codes are defined in Appendix A.11.5, "Selector Unit Control Selectors".

9.12.1. Selector Control

The Selector Control is used to select among one of the AVClusters, available at the inputs of the Selector Unit. The Selector value that shall be used to select a specific AVCluster is equal to the Input Pin Number of the Input Pin on which the desired AVCluster enters the Selector Unit. Input Pins are numbered from 1 up to total number of Input Pins of the Selector Unit. Therefore, the range of the Selector Control is implied as follows and is not advertised through the AVDD:

- The `<min>` value is always one (1)
- The `<res>` value is always one (1)
- The `<max>` value is implied by the number of `<inputPin>` (child) elements of the `<selectorInputPins>` element as advertised in the AVDD.

All Selector Units shall support this AVControl.

Table 9-14: Selector Control Table

Property	RW	L	Valid Range
CUR	RoW	M	-
NEXT	RW	O	-
<range>	**<min>**		1 (0x00000001)
	<max>		implied
	<res>		1 (0x00000001)
Field			**Value**
CS			SU_SELECTOR
IPN-ICN-OCN			0-0-0

9.13. Feature Unit Controls

The following paragraphs present a detailed description of all possible AVControls a Feature Unit can incorporate. For each AVControl, the supported Properties and their value ranges are specified. Also, the appropriate Control Selector value and the IPN-ICN-OCN values are listed. The Control Selector codes are defined in Appendix A.11.6, "Feature Unit Control Selectors".

9.13.1. VideoControls

9.13.1.1. VideoTrack Selector Control

See Section 9.7.2, "VideoTrack Selector Control" for a detailed description. "XX" in that description shall be replaced by "FU".

9.13.1.2. Brightness Control

The Brightness Control is used to adjust the brightness of a particular VideoChannel. This is a relative value where increasing values indicate increasing brightness. The actual brightness level, associated with a Brightness Control value, is vendor-dependent.

Support for this AVControl is optional.

Table 9-15: Brightness Control Table

Property	RW	L	Valid Range
CUR	RoW	M	-
NEXT	RW	O	-
<range>*	**<min>**		[-32,768.00 (0x8000.0000) – 32767.9999847 (0x7FFF.FFFF)]
	<max>		[-32,768.00 (0x8000.0000) – 32767.9999847 (0x7FFF.FFFF)]
	<res>		[0.000015258789062 (0x0000.0001) – 32767.9999847 (0x7FFF.FFFF)]
<valueList>?			[-32,768.00 (0x8000.0000) – 32767.9999847 (0x7FFF.FFFF)]
Field			**Value**
CS			FU_BRIGHTNESS
IPN-ICN-OCN			1-Channel#-Channel#

9.13.1.3. Contrast Control

The Contrast Control is used to adjust the contrast of a particular VideoChannel. This is a relative value where increasing values indicate increasing contrast. The actual contrast level, associated with a Contrast Control value, is vendor-dependent.

Support for this AVControl is optional.

Table 9-16: Contrast Control Table

Property	RW	L	Valid Range
CUR	RoW	M	-
NEXT	RW	O	-
<range>*	**<min>**		[-32,768.00 (0x8000.0000) – 32767.9999847 (0x7FFF.FFFF)]
	<max>		[-32,768.00 (0x8000.0000) – 32767.9999847 (0x7FFF.FFFF)]
	<res>		[0.000015258789062 (0x0000.0001) – 32767.9999847 (0x7FFF.FFFF)]
<valueList>?			[-32,768.00 (0x8000.0000) – 32767.9999847 (0x7FFF.FFFF)]
Field			**Value**
CS			FU_CONTRAST
IPN-ICN-OCN			1-Channel#-Channel#

9.13.2. AudioControls

9.13.2.1. AudioTrack Selector Control

See Section 9.7.3, "AudioTrack Selector Control" for a detailed description. "XX" in that description shall be replaced by "FU".

9.13.2.2. Mute Control

The Mute Control is used to temporarily suppress (mute) the sound on a particular AudioChannel.

Support for this AVControl is optional.

Table 9-17: Mute Control Table

Property	RW	L	Valid Range
CUR	RoW	M	FALSE/TRUE (0x00000000/0x00000001) – FALSE=Not Muted; TRUE=Muted.
NEXT	RW	O	FALSE/TRUE (0x00000000/0x00000001)
Field			**Value**
CS			FU_MUTE
IPN-ICN-OCN			1-Channel#-Channel#

9.13.2.3. Volume Control

The Volume Control is used to control the sound level on a particular AudioChannel.

Support for this AVControl is optional.

Table 9-18: Volume Control Table

Property	RW	L	Valid Range
CUR	RoW	M	-
NEXT	RW	O	-
<range>*	**<min>**		[-32,768.00 dB (0x8000.0000) – 32767.9999847 dB (0x7FFF.FFFF)]
	<max>		[-32,768.00 dB (0x8000.0000) – 32767.9999847 dB (0x7FFF.FFFF)]
	<res>		[0.000015258789062 dB (0x0000.0001) – 32767.9999847 dB (0x7FFF.FFFF)]
<valueList>?			[-32,768.00 dB (0x8000.0000) – 32767.9999847 dB (0x7FFF.FFFF)]
Field			**Value**
CS			FU_VOLUME
IPN-ICN-OCN			1-Channel#-Channel#

9.13.2.4. Bass Control

The Bass Control controls the bass level on a particular AudioChannel. Other parameters that also influence the bass behavior, such as cut-off frequency, cannot be altered through this Control Sequence.

Support for this AVControl is optional.

Table 9-19: Bass Control Table

Property	RW	L	Valid Range
CUR	RoW	M	-
NEXT	RW	O	-
<range>*	**<min>**		[-32,768.00 dB (0x8000.0000) – 32767.9999847 dB (0x7FFF.FFFF)]
	<max>		[-32,768.00 dB (0x8000.0000) – 32767.9999847 dB (0x7FFF.FFFF)]
	<res>		[0.000015258789062 dB (0x0000.0001) – 32767.9999847 dB (0x7FFF.FFFF)]
<valueList>?			[-32,768.00 dB (0x8000.0000) – 32767.9999847 dB (0x7FFF.FFFF)]
Field			**Value**
CS			FU_BASS
IPN-ICN-OCN			1-Channel#-Channel#

9.13.2.5. Mid Control

The Mid Control controls the mid level on a particular AudioChannel. Other parameters that also influence the mid behavior, such as cut-off frequency, cannot be altered through this Control Sequence.

Support for this AVControl is optional.

Table 9-20: Mid Control Table

Property	RW	L	Valid Range
CUR	RoW	M	-
NEXT	RW	O	-
<range>*	**<min>**		[-32,768.00 dB (0x8000.0000) – 32767.9999847 dB (0x7FFF.FFFF)]
	<max>		[-32,768.00 dB (0x8000.0000) – 32767.9999847 dB (0x7FFF.FFFF)]
	<res>		[0.000015258789062 dB (0x0000.0001) – 32767.9999847 dB (0x7FFF.FFFF)]
<valueList>?			[-32,768.00 dB (0x8000.0000) – 32767.9999847 dB (0x7FFF.FFFF)]
Field			**Value**
CS			FU_MID
IPN-ICN-OCN			1-Channel#-Channel#

9.13.2.6. Treble Control

The Treble Control controls the treble behavior on a particular AudioChannel. Other parameters that also influence the behavior of the Treble Control, such as cut-off frequency, cannot be altered through this Control Sequence.

Support for this AVControl is optional.

Table 9-21: Treble Control Table

Property	RW	L	Valid Range
CUR	RoW	M	-
NEXT	RW	O	-
<range>*	**<min>**		[-32,768.00 dB (0x8000.0000) – 32767.9999847 dB (0x7FFF.FFFF)]
	<max>		[-32,768.00 dB (0x8000.0000) – 32767.9999847 dB (0x7FFF.FFFF)]
	<res>		[0.000015258789062 dB (0x0000.0001) – 32767.9999847 dB (0x7FFF.FFFF)]
<valueList>?			[-32,768.00 dB (0x8000.0000) – 32767.9999847 dB (0x7FFF.FFFF)]
Field			**Value**
CS			FU_TREBLE
IPN-ICN-OCN			1-Channel#-Channel#

9.13.2.7. Graphics Equalizer Control

The AV Device Class Definition provides support for a graphic equalizer. The number of bands that can be supported is virtually unlimited.

In most cases, a ⅓-octave or octave band equalizer is implemented. ⅓-octave bands may be defined according to [ANSIS1_11]. Bands are numbered from 14 (center frequency of 25 Hz) up to 43 (center frequency of 20,000 Hz), making a total of 30 bands. The following table lists the band numbers and their center frequencies.

Table 9-22: Band Numbers and Center Frequencies (ANSI S1.11-1986 Standard)

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

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

A Feature Unit that supports the Graphic Equalizer Control is not required to implement either a ⅓-octave or an octave filter set. Any number of bands with any center frequency may be implemented. The exact configuration of the Graphic Equalizer is advertised via the <graphicEQ> element in the Feature Unit AVDD Description. The number of <frequency> child elements of the <filters> element indicates the total number of bands of the Graphic Equalizer where each <frequency> element advertises the center frequency of a particular band. The center frequency is expressed as an Unsigned Fixed Point (16.16) number in the range [0.00 (0x0000.0000) – 65,535.9999847 (0xFFFF.FFFF)].

Each band is assigned a BandID value based on the order in which the band's corresponding <frequency> element appears under the <filters> element. The first band is assigned BandID value #1 and the last band is assigned BandID value #n in a list of n bands.

A Graphic Equalizer Control contains one Level Control for each band it supports on a per- AudioChannel basis.

The OCN field is repurposed to contain the targeted BandID number (BandID#).

All Level Controls shall have identical <ranges> Properties.

Support for this AVControl is optional.

Table 9-23: Level Control Table

Property	RW	L	Valid Range
CUR	RoW	M	-
NEXT	RW	O	-
<range>*	**<min>**		[-32,768.00 dB (0x8000.0000) – 32767.9999847 dB (0x7FFF.FFFF)]
	<max>		[-32,768.00 dB (0x8000.0000) – 32767.9999847 dB (0x7FFF.FFFF)]
	<res>		[0.000015258789062 dB (0x0000.0001) – 32767.9999847 dB (0x7FFF.FFFF)]
<valueList>?			[-32,768.00 dB (0x8000.0000) – 32767.9999847 dB (0x7FFF.FFFF)]
Field			**Value**
CS			FU_GRAPHIC_EQ
IPN-ICN-OCN			1-Channel#-BandID#

9.13.2.8. Delay Control

The Delay Control is used to introduce a certain amount of delay on a particular AudioChannel.

Support for this AVControl is optional.

Table 9-24: Delay Control Table

Property	RW	L	Valid Range
CUR	RoW	M	-
NEXT	RW	O	-
<range>*	**<min>**		[-2,147,483,648 ns (0x80000000) – 2,147,483,647 ns (0x7FFFFFFF)]
	<max>		[-2,147,483,648 ns (0x80000000) – 2,147,483,647 ns (0x7FFFFFFF)]
	<res>		[1 ns (0x00000001) – 2,147,483,647 ns (0x7FFFFFFF)]
<valueList>?			[-2,147,483,648 ns (0x80000000) – 2,147,483,647 ns (0x7FFFFFFF)]
Field			**Value**
CS			FU_DELAY
IPN-ICN-OCN			1-Channel#-Channel#

9.13.2.9. Bass Boost Control

The Bass Boost Control is used to enable or disable an additional increase in bass level on a particular AudioChannel.

Support for this AVControl is optional.

Table 9-25: Bass Boost Control Table

Property	RW	L	Valid Range
CUR	RoW	M	FALSE/TRUE (0x00000000/0x00000001) – FALSE=Disabled; TRUE=Enabled.
NEXT	RW	O	FALSE/TRUE (0x00000000/0x00000001)
Field			**Value**
CS			FU_BASS_BOOST
IPN-ICN-OCN			1-Channel#-Channel#

9.13.2.10. Loudness Control

The Loudness Control is used to enable or disable an additional increase in loudness on a particular AudioChannel.

Support for this AVControl is optional.

Table 9-26: Loudness Control Table

Property	RW	L	Valid Range
CUR	RoW	M	FALSE/TRUE (0x00000000/0x00000001) – FALSE=Disabled; TRUE=Enabled.
NEXT	RW	O	FALSE/TRUE (0x00000000/0x00000001)
Field			**Value**
CS			FU_LOUDNESS
IPN-ICN-OCN			1-Channel#-Channel#

9.13.2.11. Input Gain Control

The Input Gain Control is used to apply attenuation or gain on a particular AudioChannel.

Support for this AVControl is optional.

Table 9-27: Input Gain Control Table

Property	RW	L	Valid Range
CUR	RoW	M	-
NEXT	RW	O	-
<range>*	**<min>**		[-32,768.00 dB (0x8000.0000) – 32767.9999847 dB (0x7FFF.FFFF)]
	<max>		[-32,768.00 dB (0x8000.0000) – 32767.9999847 dB (0x7FFF.FFFF)]
	<res>		[0.000015258789062 dB (0x0000.0001) – 32767.9999847 dB (0x7FFF.FFFF)]
<valueList>?			[-32,768.00 dB (0x8000.0000) – 32767.9999847 dB (0x7FFF.FFFF)]
Field			**Value**
CS			FU_INPUT_GAIN
IPN-ICN-OCN			1-Channel#-Channel#

9.13.2.12. Automatic Input Gain Control

The Automatic Input Gain Control (AGC) is used to enable circuitry that detects incoming sound levels and adjust input gain automatically on a particular AudioChannel. This AVControl works in concert with the Input Gain Control. When the Automatic Input Gain Control is switched on, the Input Gain Control is automatically and continuously adjusted so that outgoing sound levels are fairly constant. Attempts to programmatically adjust the Input Gain Control (see Section 9.13.2.11) result in an Error Response Message. An implementation shall not generate Notify Messages, originating from the Input Gain Control, whenever the value of that Input Gain Control changes due to an automatic adjustment. Instead, the Controller should query the Input Gain Control for its current value.

When the Automatic Input Gain is switched off, the Input Gain Control shall retain its last automatically generated setting.

Support for this AVControl is optional.

Table 9-28: Automatic Input Gain Control Table

Property	RW	L	Valid Range
CUR	RoW	M	FALSE/TRUE (0x00000000/0x00000001) – FALSE=Off; TRUE=On.
NEXT	RW	O	FALSE/TRUE (0x00000000/0x00000001)
Field			**Value**
CS			FU_AUTO_INPUT_GAIN
IPN-ICN-OCN			1-Channel#-Channel#

9.13.2.13. Input Gain Pad Control

The Input Gain Pad Control is used to apply attenuation or gain on a particular AudioChannel. The Input Gain Pad Control usually has only a few discrete (attenuation) settings (for example, 0 dB, -10 dB, -20 dB, and -30 dB).

Support for this AVControl is optional.

Table 9-29: Input Gain Pad Control Table

Property	RW	L	Valid Range
CUR	RoW	M	-
NEXT	RW	O	-
<range>*	**<min>**		[-32,768.00 dB (0x8000.0000) – 32767.9999847 dB (0x7FFF.FFFF)]
	<max>		[-32,768.00 dB (0x8000.0000) – 32767.9999847 dB (0x7FFF.FFFF)]
	<res>		[0.000015258789062 dB (0x0000.0001) – 32767.9999847 dB (0x7FFF.FFFF)]
<valueList>?			[-32,768.00 dB (0x8000.0000) – 32767.9999847 dB (0x7FFF.FFFF)]
Field			**Value**
CS			FU_INPUT_GAIN_PAD
IPN-ICN-OCN			1-Channel#-Channel#

9.13.2.14. Phase Inverter Control

The Phase Inverter Control is used to enable or disable a 180° phase shift between incoming and outgoing signal on a particular AudioChannel.

Support for this AVControl is optional.

Table 9-30: Phase Inverter Control Table

Property	RW	L	Valid Range
CUR	RoW	M	FALSE/TRUE (0x00000000/0x00000001) – FALSE=Disabled; TRUE=Enabled.
NEXT	RW	O	FALSE/TRUE (0x00000000/0x00000001)
Field			**Value**
CS			FU_AUTO_INPUT_GAIN
IPN-ICN-OCN			1-Channel#-Channel#

9.14. Effect Unit Controls

The following paragraphs present a detailed description of all possible AVControls an Effect Unit can incorporate. For each AVControl, the supported Properties and their value ranges are specified. Also, the appropriate Control Selector value and the IPN-ICN-OCN values are listed. The Control Selector codes are defined in Appendix 9.14, "Effect Unit Controls".

9.14.1. Video Effect Unit Controls

9.14.1.1. Scaling Effect Unit

This version of the specification does not provide support for this item.

9.14.2. Audio Effect Unit Controls

9.14.2.1. Parametric Equalizer Section Effect Unit

This version of the specification does not provide support for this item.

9.14.2.2. Reverberation Effect Unit

This version of the specification does not provide support for this item.

9.14.2.3. Modulation Delay Effect Unit

This version of the specification does not provide support for this item.

9.14.2.4. Dynamic Range Compressor Effect Unit

This version of the specification does not provide support for this item.

9.15. Processing Unit Controls

The following paragraphs present a detailed description of all possible AVControls a Processing Unit can incorporate. For each AVControl, the supported Properties and their value ranges are specified. Also, the appropriate Control Selector value and the IPN-ICN-OCN values are listed. The Control Selector codes are defined in Appendix A.11.8, "Processing Unit Control Selectors".

9.15.1. Video Processing Unit Controls

9.15.1.1. Still Image Processing Unit

This version of the specification does not provide support for this item.

9.15.2. Audio Processing Unit Controls

9.15.2.1. Stereo Widening Processing Unit

This version of the specification does not provide support for this item.

9.16. Converter Unit Controls

The following paragraphs present a detailed description of all possible AVControls a Converter Unit can incorporate. For each AVControl, the supported Properties and their value ranges are specified. Also, the appropriate Control Selector value and the IPN-ICN-OCN values are listed. The Control Selector codes are defined in Appendix 9.16, "Converter Unit Controls".

9.16.1. General Controls

9.16.1.1. Cluster Control

All Converter Units shall support this AVControl.

See Section 9.7.1, "Cluster Control" for a detailed description. "XX" in that description shall be replaced by "CU".

9.16.2. VideoControls

9.16.2.1. VideoTrack Selector Control

See Section 9.7.2, "VideoTrack Selector Control" for a detailed description. "XX" in that description shall be replaced by "CU".

9.16.2.2. Video Mode Selector Control

The Video Mode Selector Control is used to indicate or select an operational mode of the video part of the Converter Unit. The list of supported Video Modes is exposed through the `<videoModes>` element in the AVDD. The value that shall be used to select a specific Video Mode is derived from the order in which the supported Video Modes appear under the `<videoModes>` element. The first Video Mode is assigned index value 1; the last Video Mode is assigned the value n in a list of n Video Modes. Therefore, the range of the Video Mode Selector Control is implied as follows and is not advertised through the AVDD:

- The `<min>` value is always one (1)
- The `<res>` value is always one (1)
- The `<max>` value is implied by the number of (child) elements in the list of `<videoModes>` as advertised in the AVDD.

All Converter Units that are able to convert VideoStreams shall support this AVControl.

Table 9-31: Video Mode Selector Control Table

Property	RW	L	Valid Range
CUR	RoW	M	-
NEXT	RW	O	-
<range>	**<min>**		1 (0x00000001)
	<max>		implied
	<res>		1 (0x00000001)
Field			**Value**
CS			CU_VIDEO_MODE_SELECTOR
IPN-ICN-OCN			0-0-0

9.16.3. AudioControls

9.16.3.1. AudioTrack Selector Control

See Section 9.7.3, "AudioTrack Selector Control" for a detailed description. "XX" in that description shall be replaced by "CU".

9.16.3.2. Audio Mode Selector Control

The Audio Mode Selector Control is used to indicate or select an operational mode of the audio part of the Converter Unit. The list of supported Audio Modes is exposed through the <audioModes> element in the AVDD. The value that shall be used to select a specific Audio Mode is derived from the order in which the supported Audio Modes appear under the <audioModes> element. The first Audio Mode is assigned index value 1; the last Audio Mode is assigned the value n in a list of n Audio Modes. Therefore, the range of the Audio Mode Selector Control is implied as follows and is not advertised through the AVDD:

- The <min> value is always one (1)
- The <res> value is always one (1)
- The <max> value is implied by the number of (child) elements in the list of <audioModes> as advertised in the AVDD.

All Converter Units that are able to convert AudioStreams shall support this AVControl.

Table 9-32: Audio Mode Selector Control Table

Property	RW	L	Valid Range
CUR	RoW	M	-
NEXT	RW	O	-
<range>	**<min>**		1 (0x00000001)
	<max>		implied
	<res>		1 (0x00000001)
Field			**Value**
CS			CU_AUDIO_MODE_SELECTOR
IPN-ICN-OCN			0-0-0

9.17. Router Unit Controls

The following paragraphs present a detailed description of all possible AVControls a Router Unit can incorporate. For each AVControl, the supported Properties and their value ranges are specified. Also, the appropriate Control Selector value and the IPN-ICN-OCN values are listed. The Control Selector codes are defined in Appendix A.11.10, "Router Unit Control Selectors".

9.17.1. VideoTrack Selector Control

See Section 9.7.2, "VideoTrack Selector Control" for a detailed description. "XX" in that description shall be replaced by "RU".

9.17.2. AudioTrack Selector Control

See Section 9.7.3, "AudioTrack Selector Control" for a detailed description. "XX" in that description shall be replaced by "RU".

9.17.3. MetadataTrack Selector Control

See Section 9.7.4, "MetadataTrack Selector Control" for a detailed description. "XX" in that description shall be replaced by "RU".

9.17.4. Video Input Pin Selector Control

The Video Input Pin Selector Control is used to indicate or select the Input Pin from which the Active VideoTrack is used for the targeted VideoTrack in the outgoing VideoCluster.

The OCN field is repurposed to contain the targeted VideoTrack number (VideoTrack#).

All Router Units that are able to route VideoStreams shall support this AVControl.

Table 9-33: Video Input Pin Selector Control Table

Property	RW	L	Valid Range
CUR	RoW	M	-
NEXT	RW	O	-
<range>*	**<min>**		[1 (0x00000001) – 65,535 (0x0000FFFF)]
	<max>		[1 (0x00000001) – 65,535 (0x0000FFFF)]
	<res>		[1 (0x00000001) – 65,535 (0x0000FFFF)]
<valueList>?			[1 (0x00000001) – 65,535 (0x0000FFFF)]
Field			**Value**
CS			RU_VIDEO_INPUT_PIN_SELECTOR
IPN-ICN-OCN			0-0-VideoTrack#

9.17.5. Audio Input Pin Selector Control

The Audio Input Pin Selector Control is used to indicate or select the Input Pin from which the Active AudioTrack is used for the targeted AudioTrack in the outgoing AudioCluster.

The OCN field is repurposed to contain the targeted AudioTrack number (AudioTrack#).

All Router Units that are able to route AudioStreams shall support this AVControl.

Table 9-34: Audio Input Pin Selector Control Table

Property	RW	L	Valid Range
CUR	RoW	M	-
NEXT	RW	O	-
<range>*	**<min>**		[1 (0x00000001) – 65,535 (0x0000FFFF)]
	<max>		[1 (0x00000001) – 65,535 (0x0000FFFF)]
	<res>		[1 (0x00000001) – 65,535 (0x0000FFFF)]
<valueList>?			[1 (0x00000001) – 65,535 (0x0000FFFF)]
Field			**Value**
CS			RU_AUDIO_INPUT_PIN_SELECTOR
IPN-ICN-OCN			0-0-AudioTrack#

9.17.6. Metadata Input Pin Selector Control

The Metadata Input Pin Selector Control is used to indicate or select the Input Pin from which the Active MetadataTrack is used for the targeted MetadataTrack in the outgoing MetadataCluster.

The OCN field is repurposed to contain the targeted MetadataTrack number (MetadataTrack#).

All Router Units that are able to route MetadataStreams shall support this AVControl.

Table 9-35: Metadata Input Pin Selector Control Table

Property	RW	L	Valid Range
CUR	RoW	M	-
NEXT	RW	O	-
<range>*	<min>		[1 (0x00000001) – 65,535 (0x0000FFFF)]
	<max>		[1 (0x00000001) – 65,535 (0x0000FFFF)]
	<res>		[1 (0x00000001) – 65,535 (0x0000FFFF)]
<valueList>?			[1 (0x00000001) – 65,535 (0x0000FFFF)]
Field			**Value**
CS			RU_METADATA_INPUT_PIN_SELECTOR
IPN-ICN-OCN			0-0-MetadataTrack#

9.17.7. Cluster Control

All Router Units shall support this AVControl.

See Section 9.7.1, “Cluster Control” for a detailed description. “XX” in that description shall be replaced by “RU”.

Note: The Router Unit does not explicitly create a new outgoing AVCluster but rather takes VideoTracks, AudioTracks, and MetadataTracks from the incoming AVClusters and recombines them into the outgoing AVCluster, it would therefore be possible for Controller software to derive the outgoing AVCluster from the incoming AVCluster configurations and from the positions of the various Track Selector and Input Pin Selector Controls. However, the Cluster Control is provided to relieve the Controller from the task of assembling the outgoing AVCluster Description and providing the Controller with a single point where the AVCluster Description can be retrieved.

9.18. AVData Controls

The following paragraphs present a detailed description of all possible AVControls an AVData Entity can incorporate. For each AVControl, the supported Properties and their value ranges are specified. Also, the appropriate Control Selector value and the IPN-ICN-OCN values are listed. The Control Selector codes are defined in Appendix A.11.11, “AVData Control Selectors”.

9.18.1. Common AVData Controls

9.18.1.1. Connectors Control

The Connectors Control is a list of one or more Connector Controls as described in the following section. All the Connector Controls in a Connectors Control list have an interdependency and typically all the Connector Controls together provide one external interface to the AVFunction. Examples include:

- Left and Right Analog Line-in connectors
- Component Video connectors (Y, Cb, Cr)
- A combination of the above
- Other

The Connectors Description in the AVDD contains an ordered list of all the connectors associated with the AVData Entity, implicitly assigning a Connector Number to each Connector. For AVData Entities that only report one Connector, the Connector Number shall be set to one.

9.18.1.1.1. Connector Control

The Connector Control is used to examine the insertion state of a Connector that is associated with the AVData Entity. The CUR Property returns FALSE when no Connector is currently inserted and TRUE when a Connector is currently inserted. The Connector Control shall generate a Notify Message whenever the insertion state of the Connector changes. The Connector Control can also be queried via a GET Control Sequence at all times to retrieve its state.

The OCN field is repurposed to contain the Connector Number (Connector#).

Support for this AVControl is optional.

Table 9-36: Connector Control Table

Property	RW	L	Valid Range
CUR	R	M	FALSE/TRUE (0x00000000/0x00000001) – FALSE= No Connector Inserted; TRUE=Connector Inserted.
NEXT	-	P	-
Field			**Value**
CS			AD_CONNECTOR
IPN-ICN-OCN			0-0-Connector#

9.18.1.2. Overload Control

The Overload Control is used to indicate the existence of an overload condition within an AVData Entity. (For example, overvoltage at the analog line-in connector; thermal overload in powered speakers; etc.) Whenever an overload condition occurs, the Overload Control generates a Notify Message. The overload condition is a 'sticky' condition meaning that the occurrence of the first overload condition is memorized and subsequent overload conditions do not generate additional Notify Messages. Explicit Controller intervention is required to release the memorized overload condition. Issuing a GET Control Sequence after the Notify Message has been received releases the memorized overload condition and returns the current live state of the overload condition in the CUR Property. If an overload condition is present at the time the GET Control Sequence is processed, the CUR Property returns the value one and a new Notify Message is generated at this time. The Overload Control can also be queried via a GET Control Sequence at all times to retrieve its current state.

Support for this AVControl is optional.

Table 9-37: Overload Control Table

Property	RW	L	Valid Range
CUR	R	M	FALSE/TRUE (0x00000000/0x00000001) – FALSE=Normal; TRUE=Overload Condition Exists.
NEXT	-	P	-
Field			**Value**
CS			AD_OVERLOAD
IPN-ICN-OCN			0-0-0

9.18.1.3. Active Alternate Setting Control

The Active Alternate Setting Control is used to change the Active Alternate Setting of an AVData Entity. Controller software needs to determine which Alternate Settings of the AVData Entity to select, primarily based on the bandwidth requirements of the currently selected AVStream Configuration.

This specification does not allow an AVData Entity to change from one Active Alternate Setting to another Active Alternate Setting without Controller intervention during normal operation. (After power-up, bus reset, selection of a new configuration of the USB Device containing the AVFunction, or after the AVControl Interface is switched to or from Alternate Setting 0 an AVData Entity is required to switch to Alternate Setting 0 without Controller intervention.)

When switching Alternate Settings, Controller software should first idle all streaming activity involving the AVData Entity. The Controller shall first switch to Alternate Setting 0 (disabled) and then to the desired Active Alternate Setting. The Controller shall always wait for a successful Response Message from the Active Alternate Setting Control, indicating that the AVData Entity is ready to accept new streaming interaction.

The Active Alternate Setting Control always provides the currently Active Alternate Setting for the AVData Entity when read.

Alternate Settings are numbered contiguously from 0 up to 255. Alternate Setting 0 is the idle setting and shall be supported. In addition to Alternate Setting 0, a least one Active Alternate Setting 1 shall be supported for the AVData Entity. The total number of Alternate Settings supported and therefore the range of the AVControl can be derived from the AVData Entity Description. The `<min>` value is always 0 and the `<max>` value equals the number of `<alternateSetting>` elements in the AVData Description for that AVData Entity. Therefore, the total number of supported Alternate Settings equals [#`<alternateSetting>` elements] + 1. The +1 accounts for the idle Alternate Setting 0. Since there is no information to be conveyed for Alternate Setting 0, there is no `<alternateSetting>` element associated with Alternate Setting 0 in the AVDD.

Note: For AVData Entities other than AVData Streaming interfaces, it is most likely enough to support only two Alternate Settings, the disabled Alternate Setting 0 and one Active Alternate Setting to enable streaming. For AVData Streaming Interfaces, Alternate Settings are used to adjust isochronous USB bandwidth requirements, potentially creating the necessity for multiple Active Alternate Settings.

All AVData Entities shall support this AVControl, except for the AVData Streaming Interface Entities (see Section 9.18.1.3.1, “Special Considerations for AVData Streaming Interfaces” below for details).

Table 9-38: Active Alternate Setting Control Table

Property	RW	L	Valid Range
CUR	RW	M	-
NEXT	RW	O	-
<range>	**<min>**		0 (0x00000000)
	<max>		[1 (0x00000001) – 255 (0x000000FF)]
	<res>		1 (0x00000001)
Field			**Value**
CS			AD_ACT_ALT_SETTING
IPN-ICN-OCN			0-0-0

9.18.1.3.1. Special Considerations for AVData Streaming Interfaces

There already exist standard USB requests to manipulate Alternate Settings of a USB interface. The standard Get/Set Interface requests are used for this purpose. This specification requires that the standard Set Interface request be used to switch the Alternate Setting of an AVData Streaming Interface. Likewise, the standard Get Interface request can be used to retrieve the current Alternate Setting of an AVData Streaming Interface.

9.18.1.4. Tunnel Control

The Tunnel Control is used to send unspecified information to or receive unspecified information from an AVData Entity. The AVData Entity is not supposed to interpret the (bi-directional) byte streams but rather exchange the byte streams with the physical object the AVData Entity represents.

The Controller can send information to the Tunnel Control by issuing a SET Control Sequence. The Response Message of the SET Control Sequence indicates whether the data bytes have been successfully received (E=0) or not (E=1) by the AVData Entity. It does not indicate whether the data bytes have been successfully received and/or acted upon by the receiving physical object. It is assumed that appropriate protocols are built into the data byte streams to indicate error cases. The SETR Control Sequence shall not be supported.

The Tunnel Control shall issue a Notify Message to the Controller whenever new information is available in the AVData Entity. The new information shall be contained in that Notify Message. The GET Control Sequence shall not be supported.

Support for this AVControl is optional.

Table 9-39: Tunnel Control Table

Property	RW	L	Valid Range		
CUR	RW	M	-		
NEXT	-	P	-		
Field			**Value**		
CS			AD_TUNNEL		
IPN-ICN-OCN			0-0-0		
PB	**DATALEN**		n		
Offset	**Field**	**Size**	**Value**	**Description**	
0	**TunnelData**	n	-	The data bytes that are either sent to the AVData Entity (SET Control Sequence) or received from the AVData Entity (NOTIF Control Sequence).	

9.18.1.5. EDID Control

The EDID Control is used to retrieve the Extended Display Identification Data (EDID) from an AVData In Entity. This AVControl shall be implemented as Read-Only (only the GET Control Sequence is supported). The Controller shall specify the offset (in bytes) from which the EDID information needs to be retrieved in the **wEDIDOffset** field of the Command Message. Likewise, the Controller shall specify the number of EDID data bytes to return, starting from the

offset position as indicated by the **wEDIDOffset** field, in the **wEDIDLength** field. If the total number of bytes containing the EDID information is unknown, the Controller can specify a value of zero in the **wEDIDLength** field indicating that all remaining EDID data bytes, starting from the offset position, shall be returned in the Response Message.

Note: The AVFunction should guarantee consistency between information returned in the EDID data bytes and the equivalent information returned in the `<videoBulkStreamConfigList>` element for AVData FrameBuffer Entities and the `<videoIsoStreamConfigList>` element for AVData Video Streaming Interfaces in the AVData Description. At the very least, there should be no conflicting information. However, the information returned in the `<VideoBulkStreamConfigList>` or `<VideoIsoStreamConfigList>` element may be a superset of the currently available EDID information. This can happen when the AVData In Entity connects downstream to a detachable display and therefore (in the absence of extensive scaling facilities) not all advertised VideoStream Configurations may be valid at all times, depending on the supported resolutions and VideoFrame Rates of the attached display. It is therefore necessary for Controller software to always inspect both the advertised VideoStream Configurations and the current EDID information and find the intersection of both to determine which VideoStream Configurations are currently valid and selectable.
Whenever the EDID information changes for whatever reason, the EDID Control shall generate a Notify Message. If the change also affects the equivalent information returned in the AVData Description, then the AVDD Info Control shall generate a Notify Message, informing the Controller that an updated AV Document Description (AVDD) is available.

All AVData FrameBuffer-In Entities and AVData Video Streaming-In Interfaces shall support this AVControl.

The Control table for the Command Message of the EDID GET Control Sequence is as follows:

Table 9-40: EDID Command Control Table

Property	**RW**	**L**	**Valid Range**		
CUR	R	M	-		
NEXT	-	P	-		
Field			**Value**		
CS			AD_EDID		
IPN-ICN-OCN			0-0-0		
PB	**DATALEN**		4		
Offset	**Field**	**Size**	**Value**	**Description**	
0	**wEDIDOffset**	2	Number	The offset into the EDID information from which to retrieve the EDID data bytes.	
2	**wEDIDLength**	2	Number	The number of EDID data bytes to return, starting from the offset position as indicated by the **wEDIDOffset** field.	

The Control table for the Response Message of the EDID GET Control Sequence is as follows where n is always less or equal to the value specified in the **wEDIDLength** field of the Command Message:

Table 9-41: EDID Response Control Table

Property	RW	L	Valid Range		
CUR	R	M	-		
NEXT	-	P	-		
Field			**Value**		
CS			AD_EDID		
IPN-ICN-OCN			0-0-0		
PB	**DATALEN**		n		
Offset	**Field**	**Size**	**Value**	**Description**	
0	Data	n	-	The requested EDID data bytes.	

9.18.1.6. Stream Selector Control

The Stream Selector Control is used to indicate or select among one of the supported AVStream Configurations for the AVData Entity. The list of supported VideoStream Configurations is exposed through the `<videoBulkStreamConfigList>` element for AVData FrameBuffer Entities and the `<videoIsoStreamConfigList>` element for AVData Video Streaming Interfaces in the AVDD. The list of supported AudioStream Configurations is exposed through the`<audioStreamConfigList>` element for AVData Audio Streaming Interfaces. The Stream Selector value that shall be used to select a specific Video- or AudioStream Configuration is derived from the order in which the supported Video- or AudioStream Configurations appear under the `<videoBulkStreamConfigList>`, `<videoIsoStreamConfigList>`, or `<audioStreamConfigList>` elements. The first Video- or AudioStream Configuration is assigned index value 1; the last Video- or AudioStream Configuration is assigned the value n in a list of n Video- or AudioStream Configurations. Therefore, the range of the Stream Selector Control is implied as follows and is not advertised through the AVDD:

- The `<min>` value is always one (1)
- The `<res>` value is always one (1)
- The `<max>` value is implied by the number of child elements of the `<videoBulkStreamConfigList>`, `<videoIsoStreamConfigList>`, or `<audioStreamConfigList>` elements as advertised in the AVDD.

The Stream Selector Control may be implemented as Read-Only. This indicates to the Controller that the AVFunction performs AVStream selection independently through external means. For example, an AVFunction that has provisions to connect an external source of AV content to it may decide what AVStream Configuration it produces on its AVData FrameBuffer-Out Entity, depending on the currently connected external source.

A Read-Only Stream Selector Control shall send a Notify Message whenever the currently selected AVStream Configuration changes.

A Read-Write Stream Selector Control requires Controller intervention to select the desired AVStream Configuration.

Note: Not all advertised AVStream Configurations may be available at all times for a particular AVData Entity, depending on conditions, external to the AVData Entity. The AVFunction shall provide enough information to the Controller to determine unambiguously which AVStream Configurations are valid at any given time for a particular AVData Entity. For example, the Controller should examine current EDID information to determine the validity of the advertised VideoStream Configurations in the case of an AVData FrameBuffer-In or AVData Video Streaming-In Entity.

All AVData Entities that are able to interact with AVStreams (AVData FrameBuffer Entities and AVData Streaming Interfaces) shall support this AVControl.

Table 9-42: Stream Selector Control Table

Property	RW	L	Valid Range
CUR	RoW	M	-
NEXT	RW	O	-
<range>	**<min>**		1 (0x00000001)
	<max>		implied
	<res>		1 (0x00000001)
Field			**Value**
CS			AD_STREAM_SELECTOR
IPN-ICN-OCN			0-0-0

9.18.1.7. Reference Clock Control

The Reference Clock Control is a read-only AVControl, used to indicate the Clock Domain Reference Clock frequency. Only AVData Entities that act as a Clock Domain Source in some or all of their Active Alternate Settings may expose a Reference Clock Control in those Alternate Settings.

For Clock Domain Source AVData Streaming Interfaces, the Reference Clock Control indicates the Reference Clock frequency from which the actual operating frequency of the interface is derived. The actual operating frequency of the Streaming Interface is indicated by the Video- or AudioFrame Rate parameter in the Video- or AudioStream Configuration that is currently selected. If the Reference Clock Control is omitted, then the Reference Clock Frequency and the operating frequency of the interface are considered to be identical.

Note: An AVData Entity that acts as Clock Domain Sink has its operating frequency expressed as a ratio compared to the Reference Clock frequency of its affiliated Clock Domain.

For Clock Domain Source AVData Entities that are not AVData Streaming Interfaces, this AVControl is required.

Whenever the Reference Clock Control changes value, a Notify Message shall be generated to inform the Controller of the change. The Reference Clock Control can also be queried via a GET Control Sequence at all times to retrieve its value.

The range of the Reference Clock Control is defined by specification as follows and is not advertised through the AVDD:

Support for this AVControl is optional.

Table 9-43: Reference Clock Control Table

Property	RW	L	Valid Range
CUR	R	M	-
NEXT	-	P	-
<range>	**<min>**		1 Hz (0x00000001)
	<max>		4,294,967,295 Hz (0xFFFFFFFF)
	<res>		[1 Hz (0x00000001)
Field			**Value**
CS			AD_ REFERENCE_CLOCK
IPN-ICN-OCN			0-0-0

9.18.1.8. Clock Connector Control

The presence of the Clock Connector Control is used to indicate that the Clock Domain Reference Clock is supplied to the AVData Entity via an external Connector. Only AVData Entities that act as a Clock Domain Source may expose a Clock Connector Control.

The Clock Connector Control is used to examine the insertion state of the Clock Connector that is associated with the AVData Entity. The CUR Property returns FALSE when no Connector is currently inserted and TRUE when a Connector is currently inserted. The Clock Connector Control shall generate a Notify Message whenever the insertion state of the Clock Connector changes. The Clock Connector Control can also be queried via a GET Control Sequence at all times to retrieve its state.

The OCN field is repurposed to contain the Clock Connector Number (Connector#).

Support for this AVControl is optional.

Table 9-44: Clock Connector Control Table

Property	RW	L	Valid Range
CUR	R	M	FALSE/TRUE (0x00000000/0x00000001) – FALSE= No Connector Inserted; TRUE=Connector Inserted.
NEXT	-	P	-
Field			**Value**
CS			AD_CLOCK_CONNECTOR
IPN-ICN-OCN			0-0-Connector#

9.18.1.9. Clock Valid Control

The Clock Valid Control is used to indicate if the clock signal that is generated by the AVData Entity (acting as a Clock Domain Source) is valid (stable and reliable). The internal clock signal may be derived in several different ways. For example, a local oscillator can be used (asynchronous), or the clock can be derived based on the incoming data stream (adaptive). In other situations, the clock may be synchronized to the USB SOF or BIB (synchronous). Whenever the Clock Valid Control changes state, a Notify Message shall be generated to inform the Controller of the change. The Clock Valid Control can also be queried via a GET Control Sequence at all times to retrieve its state.

Support for this AVControl is optional.

Table 9-45: Clock Valid Control Table

Property	RW	L	Valid Range
CUR	R	M	FALSE/TRUE (0x00000000/0x00000001) – FALSE= Invalid Clock; TRUE=Valid Clock.
NEXT	-	P	-
Field			**Value**
CS			AD_CLOCK_VALID
IPN-ICN-OCN			0-0-0

9.18.1.10. Pitch Control

The Pitch Control is used to indicate to the AVData Entity that some larger-than-usual shifts in operating frequency are to be expected. This AVControl is only allowed for an AVData Entity that has the capability to lock its operating frequency on an external stimulus (adaptive). For example, an AVData Audio Streaming Interface that drives a Clock Domain and that has an associated adaptive endpoint may infer operating frequency information from the endpoint, based on the average data rate with which the endpoint is receiving audio data from the Controller.

The Controller uses the Pitch Control to inform the AVData Entity Entity that the actual operating frequency is going to deliberately vary more than in the normal steady state situation. The AVData Entity may react to this by switching to a coarser frequency-tracking algorithm than the one it uses during normal steady state operation.

Support for this AVControl is optional.

Table 9-46: Pitch Control Table

Property	RW	L	Valid Range
CUR	RoW	M	FALSE/TRUE (0x00000000/0x00000001) – FALSE=Normal; TRUE=Expect shifts in operating frequency.
NEXT	-	P	-
Field			**Value**
CS			AD_PITCH
IPN-ICN-OCN			0-0-0

9.18.1.11. Decoder Controls

This version of the specification does not provide support for this item.

9.18.1.12. Encoder Controls

This version of the specification does not provide support for this item.

9.18.2. AVData In Controls

9.18.2.1. AVData Generic-In Controls

9.18.2.1.1. Connectors Control

See Section 9.18.1.1, “Connectors Control” for a detailed description.

9.18.2.1.2. Overload Control

See Section 9.18.1.2, “Overload Control” for a detailed description.

9.18.2.1.3. Active Alternate Setting Control

See Section 9.18.1.3, “Active Alternate Setting Control” for a detailed description.

9.18.2.1.4. Tunnel Control

See Section 9.18.1.4, “Tunnel Control” for a detailed description.

9.18.2.1.5. Reference Clock Control

See Section 9.18.1.7, “Reference Clock Control” for a detailed description.

9.18.2.1.6. Clock Connector Control

See Section 9.18.1.8, “Clock Connector Control” for a detailed description.

9.18.2.1.7. Clock Valid Control

See Section 9.18.1.9, “Clock Valid Control” for a detailed description.

9.18.2.1.8. Pitch Control

See Section 9.18.1.10, “Pitch Control” for a detailed description.

9.18.2.2. AVData FrameBuffer-In Controls

9.18.2.2.1. Active Alternate Setting Control

See Section 9.18.1.3, “Active Alternate Setting Control” for a detailed description.

9.18.2.2.2. Tunnel Control

See Section 9.18.1.4, “Tunnel Control” for a detailed description.

9.18.2.2.3. EDID Control

See Section 9.18.1.5 “EDID Control” for a detailed description.

9.18.2.2.4. SourceData Control

The SourceData Control is used to download updated frame information to the AVData FrameBuffer-In Entity.

The format of the actual data is described and controlled by a separate [AVFORMAT_1] document. Its supported formats can range from uncompressed pixel data for the entire bitmap of the video frame to partial frame updates to even embedded compressed partial frame updates. The frame data can be multi-channel. The SourceData Control provides an encapsulation mechanism to deliver the properly formatted data from the Controller to the AVData FrameBuffer-In Entity.

Note that this AVControl is usually a Write-Only (W) Control. However, implementations may choose to make the SourceData Control readable as well (WoR – Write with optional Read).

All AVData FrameBuffer-In Entities shall support this AVControl.

Table 9-47: SourceData Control Table

Property	RW	L	Valid Range		
CUR	WoR	M	-		
NEXT	-	P	-		
Field			**Value**		
CS			AD_SOURCEDATA		
IPN-ICN-OCN			0-0-0		
PB	**DATALEN**		n		
Offset	**Field**	**Size**	**Value**	**Description**	
0	Data	n	-	The actual data that is used by the SourceData Control to generate one video frame and optionally, the associated audio stream for the duration of the frame.	

9.18.2.2.5. Stream Selector Control

See Section 9.18.1.6, “Stream Selector Control” for a detailed description.

9.18.2.2.6. Reference Clock Control

See Section 9.18.1.7, “Reference Clock Control” for a detailed description.

9.18.2.2.7. Clock Connector Control

See Section 9.18.1.8, “Clock Connector Control” for a detailed description.

9.18.2.2.8. Clock Valid Control

See Section 9.18.1.9, “Clock Valid Control” for a detailed description.

9.18.2.3. AVData Video Streaming-In Interface Controls

9.18.2.3.1. Active Alternate Setting Control

The Active Alternate Setting Control is not allowed for an AVData Video Streaming-In interface. Instead, the same functionality is achieved using the standard USB Get/Set Interface (Alternate Setting) request.

9.18.2.3.2. Tunnel Control

See Section 9.18.1.4, “Tunnel Control” for a detailed description.

9.18.2.3.3. EDID Control

See Section 9.18.1.5 “EDID Control” for a detailed description.

9.18.2.3.4. Stream Selector Control

See Section 9.18.1.6, “Stream Selector Control” for a detailed description.

9.18.2.3.5. Reference Clock Control

See Section 9.18.1.7, “Reference Clock Control” for a detailed description.

9.18.2.3.6. Clock Connector Control

See Section 9.18.1.8, “Clock Connector Control” for a detailed description.

9.18.2.3.7. Clock Valid Control

See Section 9.18.1.9, “Clock Valid Control” for a detailed description.

9.18.2.4. AVData Audio Streaming-In Interface Controls

9.18.2.4.1. Active Alternate Setting Control

The Active Alternate Setting Control is not allowed for an AVData Audio Streaming-In interface. Instead, the same functionality is achieved using the standard USB Set Interface (Alternate Setting) request.

9.18.2.4.2. Tunnel Control

See Section 9.18.1.4, “Tunnel Control” for a detailed description.

9.18.2.4.3. Audio Language Control

The Audio Language Control is used to set the language tags for the AudioTracks of an AudioBundle that is streamed from the Controller to an AVData Audio Streaming-In interface. This allows the AVFunction to add this information to the AudioCluster information, once it enters the AVCore. This AVControl shall be implemented as Read-Write (RW).

The language tag is communicated to the AVFunction in the form of a UTF-8 encoded string, and the language tag shall be expressed in accordance with [RFC5646]. This AVControl shall initialize with string value “i-default” (when read before being written).

The OCN field is repurposed to contain the targeted AudioTrack number (AudioTrack#).

Support for this AVControl is optional.

Table 9-48: Audio Language Control Table

Property	**RW**	**L**	**Valid Range**		
CUR	RW	M	-		
NEXT	RW	O	-		
Field			**Value**		
CS			AD_AUDIO_LANGUAGE		
IPN-ICN-OCN			0-0-AudioTrack#		
PB	**DATALEN**		n		
Offset	**Field**		**Size**	**Value**	**Description**
0	Data		n	String	The UTF-8-encoded language tag, expressed in accordance with [RFC5646].

9.18.2.4.4. Stream Selector Control

See Section 9.18.1.6, “Stream Selector Control” for a detailed description.

9.18.2.4.5. Reference Clock Control

See Section 9.18.1.7, “Reference Clock Control” for a detailed description.

9.18.2.4.6. Clock Connector Control

See Section 9.18.1.8, “Clock Connector Control” for a detailed description.

9.18.2.4.7. Clock Valid Control

See Section 9.18.1.9, “Clock Valid Control” for a detailed description.

9.18.2.4.8. Pitch Control

See Section 9.18.1.10, “Pitch Control” for a detailed description.

9.18.3. AVData Out Controls

9.18.3.1. AVData Generic-Out Controls

9.18.3.1.1. Connectors Control

See Section 9.18.1.1, “Connectors Control” for a detailed description.

9.18.3.1.2. Overload Control

See Section 9.18.1.2, “Overload Control” for a detailed description.

9.18.3.1.3. Active Alternate Setting Control

See Section 9.18.1.3, “Active Alternate Setting Control” for a detailed description.

9.18.3.1.4. Tunnel Control

See Section 9.18.1.4, “Tunnel Control” for a detailed description.

9.18.3.1.5. Reference Clock Control

See Section 9.18.1.7, “Reference Clock Control” for a detailed description.

9.18.3.1.6. Clock Connector Control

See Section 9.18.1.8, “Clock Connector Control” for a detailed description.

9.18.3.1.7. Clock Valid Control

See Section 9.18.1.9, “Clock Valid Control” for a detailed description.

9.18.3.1.8. Pitch Control

See Section 9.18.1.10, “Pitch Control” for a detailed description.

9.18.3.2. AVData FrameBuffer-Out Controls

9.18.3.2.1. Active Alternate Setting Control

See Section 9.18.1.3, “Active Alternate Setting Control” for a detailed description.

9.18.3.2.2. Tunnel Control

See Section 9.18.1.4, “Tunnel Control” for a detailed description.

9.18.3.2.3. SinkData Control

The SinkData Control is used to upload updated frame information from the AVData FrameBuffer-Out Entity.

The format of the actual data is described and controlled by a separate [AVFORMAT_1] document. Its supported formats can range from uncompressed pixel data for the entire bitmap of the frame via partial frame updates to even embedded compressed partial frame updates. The frame data can be multi-channel. The SinkData Control provides an encapsulation mechanism to deliver that properly formatted data packet from the AVData FrameBuffer-Out Entity to the Controller.

Note that this AVControl is always a Read-Only Control. Whenever the AVControl has a consistent updated frame buffer available, it shall send a Notify Message to the Controller, containing that updated frame buffer data, as follows:

All AVData FrameBuffer-Out Entities shall support this AVControl.

Table 9-49: SinkData Control Table

Property	RW	L	Valid Range		
CUR	R	M	-		
NEXT	-	P	-		
Field			**Value**		
CS			AD_SINKDATA		
IPN-ICN-OCN			0-0-0		
PB	**DATALEN**		n		
Offset	**Field**	**Size**	**Value**	**Description**	
0	Data	n	-	The actual data that is produced by the SinkData Control to describe one video frame.	

9.18.3.2.4. Stream Selector Control

See Section 9.18.1.6, "Stream Selector Control" for a detailed description.

9.18.3.2.5. Reference Clock Control

See Section 9.18.1.7, "Reference Clock Control" for a detailed description.

9.18.3.2.6. Clock Connector Control

See Section 9.18.1.8, "Clock Connector Control" for a detailed description.

9.18.3.2.7. Clock Valid Control

See Section 9.18.1.9, "Clock Valid Control" for a detailed description.

9.18.3.3. AVData HDMI-Out Controls

9.18.3.3.1. Connectors Control

See Section 9.18.1.1, "Connectors Control" for a detailed description.

9.18.3.3.2. Active Alternate Setting Control

See Section 9.18.1.3, "Active Alternate Setting Control" for a detailed description.

9.18.3.3.3. Tunnel Control

See Section 9.18.1.4, "Tunnel Control" for a detailed description.

9.18.3.3.4. CEC Controls

The different CEC Controls are used in tandem to gain access to the Consumer Electronics Control bus that is part of the HDMI implementation. Communication on the CEC bus is bidirectional over a single wire and data is transmitted using CEC frames. A CEC frame on the CEC bus consists of a concatenation of 10-bit words and the start of each CEC frame is indicated with a start bit. Each 10-bit word consists of 1 byte (8 bits) of data, followed by an ACK (acknowledge) bit and an EOM (end of message) bit. Only the last word of a frame will have its EOM bit set, indicating the end of the message. (For details, see [HDMI].)

The Controller communicates with the CEC Control on a data byte stream level.

This specification defines three different CEC Controls that together provide complete controlled access (both read and write) to the CEC bus. These CEC Controls are:

- CEC Write Control
- CEC Write Status Control
- CEC Read Control

If an AVData Entity supports CEC access, it shall support all three of the above CEC Controls.

The following sections outline how the CEC Controls shall be used to access the CEC bus.

9.18.3.3.4.1. Sending a CEC Frame

When sending, the Controller shall send the data bytes that make up a single CEC frame to the CEC Write Control in a single SET/SETR Control Sequence. A CEC frame is maximum 16 bytes.

It is the responsibility of the underlying software/firmware/hardware to implement the low level CEC bus protocol (manage the start bit, apply correct framing into 10 bits for each data byte and then gaining access to the CEC bus and sending the CEC message and monitoring successful transmission, doing retransmissions if necessary).

Note that this AVControl is usually a Write-Only (W) Control (only SET Control Sequence allowed). However, implementations may choose to make the CEC Write Control readable as well (WoR – Write with optional Read). The lastly written CEC frame is then available for read back, either via a SETR Control Sequence or via a GET Control Sequence.

Note that a successful completion of the SET/SETR Control Sequence only indicates that the CEC Write Control has successfully accepted the CEC message to be sent over the CEC bus, but the message may not have been sent over the CEC bus yet. It does not indicate that the CEC recipient of the CEC message has received and acknowledged the CEC message. The completion status whether the CEC recipient has received and acknowledged the CEC message is indicated by a separate Notify Message from the associated CEC Write Status Control (see below).

Table 9-50: CEC Write Control Table

Property	RW	L	Valid Range		
CUR	WoR	M	-		
NEXT	-	P	-		
Field			**Value**		
CS			AD_CEC_WRITE		
IPN-ICN-OCN			0-0-0		
PB	**DATALEN**		n		
Offset	**Field**	**Size**	**Value**	**Description**	
0	Data	n	-	The CEC frame data bytes that will be sent over the CEC bus.	

Once the CEC message has been processed and sent over the CEC bus, the CEC Write Status Control shall issue a Notify Message that indicates whether the recipient on the CEC bus has successfully received the CEC message. The format of the Notify Message is as follows:

Table 9-51: CEC Write Status Control Table

Property	RW	L	Valid Range		
CUR	R	M	-		
NEXT	-	P	-		
Field			**Value**		
CS			AD_CEC_WRITE_STATUS		
IPN-ICN-OCN			0-0-0		
PB	**DATALEN**		1		
Offset	**Field**	**Size**	**Value**	**Description**	
0	bStatus	1	Number	0: CEC frame complete and acknowledged. 1: CEC frame complete, no acknowledge. 2: Bit error, CEC frame incomplete. All other values are reserved and shall not be used.	

9.18.3.3.4.2. Receiving a CEC Frame

When the CEC Read Control receives a CEC frame from the CEC bus, it shall send a Notify Message to the Controller that contains status information about the received CEC frame and the received frame itself. It is the responsibility of

the underlying software/firmware/hardware to strip the low level CEC bus protocol and only return the actual data bytes in the CEC frame. A CEC frame cannot be retrieved through a GET Control Sequence.

Table 9-52: CEC Read Control Table

Property	RW	L	Valid Range		
CUR	R	M	-		
NEXT	-	P	-		
Field			**Value**		
CS			AD_CEC_READ		
IPN-ICN-OCN			0-0-0		
PB	**DATALEN**		n+1		
Offset	**Field**		**Size**	**Value**	**Description**
0	Status		1	Number	0: CEC frame complete and acknowledged. 1: CEC frame complete, no acknowledge. 2: Bit error, CEC frame incomplete. All other values are reserved and shall not be used.
1	Data		n	-	The CEC frame data bytes that were received over the CEC bus.

9.18.3.3.5. Reference Clock Control

See Section 9.18.1.7, “Reference Clock Control” for a detailed description.

9.18.3.3.6. Clock Connector Control

See Section 9.18.1.8, “Clock Connector Control” for a detailed description.

9.18.3.3.7. Clock Valid Control

See Section 9.18.1.9, “Clock Valid Control” for a detailed description.

9.18.3.4. AVData Video Streaming-Out Interface Controls

9.18.3.4.1. Active Alternate Setting Control

The Active Alternate Setting Control is not allowed for an AVData Video Streaming-Out interface. Instead, the same functionality is achieved using the standard USB Set Interface (Alternate Setting) request.

9.18.3.4.2. Tunnel Control

See Section 9.18.1.4, “Tunnel Control” for a detailed description.

9.18.3.4.3. Stream Selector Control

See Section 9.18.1.6, “Stream Selector Control” for a detailed description.

9.18.3.4.4. Reference Clock Control

See Section 9.18.1.7, “Reference Clock Control” for a detailed description.

9.18.3.4.5. Clock Connector Control

See Section 9.18.1.8, “Clock Connector Control” for a detailed description.

9.18.3.4.6. Clock Valid Control

See Section 9.18.1.9, “Clock Valid Control” for a detailed description.

9.18.3.5. AVData Audio Streaming-Out Interface Controls

9.18.3.5.1. Active Alternate Setting Control

The Active Alternate Setting Control is not allowed for an AVData Audio Streaming-Out interface. Instead, the same functionality is achieved using the standard USB Set Interface (Alternate Setting) request.

9.18.3.5.2. Tunnel Control

See Section 9.18.1.4, “Tunnel Control” for a detailed description.

9.18.3.5.3. Stream Selector Control

See Section 9.18.1.6, “Stream Selector Control” for a detailed description.

9.18.3.5.4. Reference Clock Control

See Section 9.18.1.7, “Reference Clock Control” for a detailed description.

9.18.3.5.5. Clock Connector Control

See Section 9.18.1.8, “Clock Connector Control” for a detailed description.

9.18.3.5.6. Clock Valid Control

See Section 9.18.1.9, “Clock Valid Control” for a detailed description.

9.18.3.5.7. Pitch Control

See Section 9.18.1.10, “Pitch Control” for a detailed description.

Appendix A. AV Device Class Codes

A.1. AVFunction Class Code

Table A-1: AVFunction Class Code

AVFunction Class Code	Value
AV_FUNCTION	AV

A.2. AVFunction Subclass Codes

Table A-2: AVFunction Subclass Codes

AVFunction Subclass Code	Value
FUNCTION_SUBCLASS_UNDEFINED	0x00

A.3. AVFunction Protocol Codes

Table A-3: AVFunction Protocol Codes

AVFunction Protocol Code	Value
FUNCTION_PROTOCOL_UNDEFINED	0x00

A.4. AV Interface Class Code

Table A-4: AV Interface Class Code

AV Interface Class Code	Value
AV	0x10 (assigned by USB)

A.5. AV Interface Subclass Codes

Table A-5: AV Interface Subclass Code

AV Interface Subclass Code	Value
INTERFACE_SUBCLASS_UNDEFINED	0x00
AVCONTROL	0x01
AVDATA_VIDEO_STREAMING	0x02
AVDATA_AUDIO_STREAMING	0x03

A.6. AV Interface Protocol Codes

Table A-6: AV Interface Protocol Codes

AV Interface Protocol Code	Value
INTERFACE_PROTOCOL_UNDEFINED	0x00
IP_VERSION_01_00	0x10

A.7. Encoder Type Codes

This version of the specification does not provide support for this item.

A.8. Decoder Type Codes

This version of the specification does not provide support for this item.

A.9. AV Request Codes

Table A-7: AV Request Codes

Control Selector	Value
REQUEST_UNDEFINED	0b0000
SETR	0b0001
SET	0b0010
GET	0b0011
NOTIF	0b0100
RESERVED	0b0110 – 0b1111

A.10. AV Property Codes

Table A-8: AV Property Codes

Property Code	Value
RESERVED	0b0000
CUR	0b0001
NEXT	0b0010
RESERVED	0b0011 – 0b1111

A.11. AVControl Selector Codes

A.11.1. AVControl Interface Control Selectors

Table A-9: AVControl Interface Control Selectors

Control Selector	Value
AC_CONTROL_UNDEFINED	0x0000
AC_AVDD_INFO	0x0001
AC_AVDD_CONTENT	0x0002
AC_COMMIT	0x0003
AC_POWER_LINE_FREQ	0x0004
AC_REMOTE_ONLY	0x0005

A.11.2. Terminal Control Selectors

Table A-10: Terminal Control Selectors

Control Selector	Value
TE_CONTROL_UNDEFINED	0x0000
TE_CLUSTER	0x0001

A.11.3. Mixer Unit Control Selectors

Table A-11: Mixer Unit Control Selectors

Control Selector	Value
MU_CONTROL_UNDEFINED	0x0000
MU_AUDIOTRACK_SELECTOR	0x0001
MU_LEVEL	0x0002

A.11.4. Metadata Unit Control Selectors

This version of the specification does not provide support for this item.

A.11.5. Selector Unit Control Selectors

Table A-12: Selector Unit Control Selectors

Control Selector	Value
SU_CONTROL_UNDEFINED	0x0000
SU_ SELECTOR	0x0001

A.11.6. Feature Unit Control Selectors

Table A-13: Feature Unit Control Selectors

Control Selector	Value
FU_CONTROL_UNDEFINED	0x0000
FU_VIDEOTRACK_SELECTOR	0x0001
FU_BRIGHTNESS	0x0002
FU_CONTRAST	0x0003
FU_AUDIOTRACK_SELECTOR	0x0004
FU_MUTE	0x0005
FU_VOLUME	0x0006
FU_BASS	0x0007
FU_MID	0x0008
FU_TREBLE	0x0009
FU_GRAPHIC_EQ	0x000A
FU_DELAY	0x000B
FU_BASS_BOOST	0x000C
FU_LOUDNESS	0x000D
FU_INPUT_GAIN	0x000E
FU_AUTO_INPUT_GAIN	0x000F
FU_INPUT_GAIN_PAD	0x0010
FU_PHASE_INVERTER	0x0011

A.11.7. Effect Unit Control Selectors

A.11.7.1. Scaling Effect Unit Control Selectors

This version of the specification does not provide support for this item.

A.11.7.2. Parametric Equalizer Section Effect Unit Control Selectors

This version of the specification does not provide support for this item.

A.11.7.3. Reverberation Effect Unit Control Selectors

This version of the specification does not provide support for this item.

A.11.7.4. Modulation Delay Effect Unit Control Selectors

This version of the specification does not provide support for this item.

A.11.7.5. Dynamic Range Compressor Effect Unit Control Selectors

This version of the specification does not provide support for this item.

A.11.8. Processing Unit Control Selectors

This version of the specification does not provide support for this item.

A.11.8.1. Still Image Processing Unit Control Selectors

This version of the specification does not provide support for this item.

A.11.8.2. Stereo Widening Processing Unit Control Selectors

This version of the specification does not provide support for this item.

A.11.9. Converter Unit Control Selectors

Table A-14: Converter Unit Control Selectors

Control Selector	Value
CU_CONTROL_UNDEFINED	0x0000
CU_VIDEOTRACK_SELECTOR	0x0001
CU_AUDIOTRACK_SELECTOR	0x0002
CU_VIDEO_MODE_SELECTOR	0x0003
CU_AUDIO_MODE_SELECTOR	0x0004
CU_CLUSTER	0x0005

A.11.10. Router Unit Control Selectors

Table A-15: Router Unit Control Selectors

Control Selector	Value
RU_CONTROL_UNDEFINED	0x0000
RU_VIDEOTRACK_SELECTOR	0x0001
RU_AUDIOTRACK_SELECTOR	0x0002
RU_METADATATRACK_SELECTOR	0x0003
RU_VIDEO_INPUT_PIN_SELECTOR	0x0004
RU_AUDIO_INPUT_PIN_SELECTOR	0x0005
RU_METADATA_INPUT_PIN_SELECTOR	0x0006
RU_CLUSTER	0x0007

A.11.11. AVData Control Selectors

Table A-16: AVData Control Selectors

Control Selector	Value
AD_CONTROL_UNDEFINED	0x0000
AD_CONNECTOR	0x0001
AD_OVERLOAD	0x0002
AD_ACT_ALT_SETTING	0x0003
AD_TUNNEL	0x0004
AD_AUDIO_LANGUAGE	0x0005
AD_CEC_WRITE	0x0006
AD_CEC_WRITE_STATUS	0x0007
AD_CEC_READ	0x0008
AD_SOURCEDATA	0x0009
AD_SINKDATA	0x000A

Control Selector	Value
AD_STREAM_SELECTOR	0x000B
AD_EDID	0x000C
AD_REFERENCE_CLOCK	0x000D
AD_CLOCK_CONNECTOR	0x000E
AD_CLOCK_VALID	0x000F
AD_PITCH	0x0010

A.12. AV Device Class General Constants

Table A-17: AV Device Class General Constants

Control Selector	Value
AV_MESSAGE_GRANULARITY	32 bytes (shall be a power of 2)
AV_MESSAGE_INVAR_SIZE	16 bytes

A.13. Legacy View Descriptor Constants

A.13.1. Descriptor Type Codes

Table A-18: Descriptor Type Codes

Descriptor Type Code	Value
AVCONTROL_IF	0x21
TERMINAL	0x22
UNIT	0x23
AVDATA	0x24
AVCONTROL	0x25
RANGES	0x26
VIDEOBULK	0x27
VIDEOISO	0x28
AUDIOISO	0x29

A.13.2. Terminal Type Codes

Table A-19: Terminal Type Codes

Terminal Type Code	Value
INPUT	0x0001
OUTPUT	0x0002

A.13.3. Unit Type Codes

Table A-20: Unit Type Codes

Unit Type Code	Value
MIXER	0x0001
SELECTOR	0x0002

Unit Type Code	Value
FEATURE	0x0003
EFFECT	0x0004
PROCESSING	0x0005
CONVERTER	0x0006
ROUTER	0x0007

A.13.4. AVData Entity Codes

Table A-21: AVData Entity Codes

AVData Entity Code	Value
GENERIC	0x0001
FRAMEBUFFER	0x0002
VIDEOSTREAMING	0x0003
AUDIOSTREAMING	0x0004
HDMI	0x0005

A.13.5. Ranges Codes

Table A-22: Ranges Codes

Ranges Type Code	Value
RANGE	0x0001
VALUELIST	0x0002

Appendix B. AVData Entity Types

B.1. Video Types

Table B-1: Video Types

Type	Identifier	Description
No Video	NONE	No video in present for this AVData Entity.
Generic	GENERIC	A generic video device.
Camera	CAMERA	A generic camera that does not fit under any of the other classifications.
Webcam	WEBCAM	A camera normally placed on the desktop or integrated into the monitor.
Personal camera	PERSONAL_CAM	A head-mounted or clip-on camera.
Display	DISPLAY	A generic display that does not fit under any of the other classifications.
PC Monitor	PC_MONITOR	A display that is to be used with a personal computer.
TV Monitor	TV_MONITOR	A TV that has the ability to also function as a display for other (external) content.
In-car display	IN_CAR_DISPLAY	A display that is built into a car.
Video Pattern Generator	PATTERN_GEN	A video device that generates a set of video test patterns.
Videophone, no echo reduction	VIDEOPHONE	The video part of a hands-free AV communication device designed for host-based echo cancellation.
Echo-suppressing videophone	ES_VIDEOPHONE	The video part of a hands-free AV communication device with echo suppression capable of half-duplex operation.
Echo-canceling videophone	EC_VIDEOPHONE	The video part of a hands-free AV communication device with echo cancellation capable of full-duplex operation.
Analog connection	ANALOG	A generic analog video connection that does not fit under any of the other classifications.
Composite connection	COMPOSITE	A composite analog video connection.
S-Video connection	S_VIDEO	An S-Video analog connection.
Component connection	COMPONENT	An analog component video connection.
Digital connection	DIGITAL	A generic digital video connection that does not fit under any of the other classifications.
DisplayPort connection	DP	The video part of a DisplayPort digital connection.
DVI connection	DVI	A DVI connection.
VGA connection	VGA	A VGA connection.
HDMI Input connection	HDMI-IN	The video part of an HDMI Input digital connection. (Note: HDMI-Out is treated as a specialised AVData Entity)
TV Tuner	TV_TUNER	Video part of TV tuner.
Satellite Receiver	SAT_TUNER	Video part of satellite receiver.
Cable Tuner	CABLE_TUNER	Video part of cable tuner.
DSS	DSS_TUNER	Video part of DSS receiver.
TV Transmitter	TV_TX	Video part of a TV transmitter
Recorder	RECORDER	A recording system.
CD player	CD	Video part of a Compact Disc player/recorder.
Compressed Video Player	COMPRESSED	Compressed Video player/recorder.
Analog Tape	ANALOG_TAPE	Video part of an Analog Video Tape Player/Recorder.
Digital Tape	DIGITAL_TAPE	Video part of a Digital Video Tape Player/Recorder.
VCR Video	VCR	Video part of a Video Cassette Player/Recorder.
Video Disc Video	VD	Video part of a VideoDisc player/recorder.

Type	Identifier	Description
DVD Video	DVD	Video part of a DVD player.
BluRay Video	BLURAY	Video part of a Blu-Ray player.
Other	OTHER	A video device other than any of the ones listed above.

B.2. Audio Types

Table B-2: Audio Types

Type	Identifier	Description
No Audio	NONE	No audio in present for this AVData Entity.
Generic	GENERIC	A generic audio input device.
Microphone	MICROPHONE	A generic microphone that does not fit under any of the other classifications.
Desktop microphone	DESKTOP_MIC	A microphone normally placed on the desktop or integrated into the monitor.
Personal microphone	PERSONAL_MIC	The microphone part of a VR head mounted display or a clip-on microphone.
Omni-directional microphone	OMNIDIRECTIONAL_MIC	A microphone designed to pick up voice from more than one speaker at relatively long ranges.
Microphone array	MIC_ARRAY	An array of microphones designed for directional processing using host-based signal processing algorithms.
Processing microphone array	PROC_MIC_ARRAY	An array of microphones with an embedded signal processor.
Speaker	SPEAKER	A generic speaker or set of speakers that does not fit under any of the other classifications.
Desktop speaker	DESKTOP_SPEAKER	Relatively small speaker or set of speakers normally placed on the desktop or integrated into the monitor. These speakers are close to the user and have limited stereo separation.
Personal speaker	PERSONAL_SPEAKER	The speaker part of a VR head mounted display.
Room speaker	ROOM_SPEAKER	Larger speaker or set of speakers that are heard well anywhere in the room.
Communication speaker	COMM_SPEAKER	Speaker or set of speakers designed for voice communication.
Low frequency effects speaker	LFE_SPEAKER	Speaker designed for low frequencies (subwoofer). Not capable of reproducing speech or music.
Headphones	HEADPHONES	A head-mounted audio output device.
Handset	HANDSET	Handheld bidirectional audio device.
Headset	HEADSET	Headmounted bidirectional audio device.
Speakerphone, no echo reduction	SPEAKERPHONE	A handsfree audio device designed for host-based echo cancellation.
Echo-suppressing speakerphone	ES_SPEAKERPHONE	A handsfree audio device with echo suppression capable of halfduplex operation.
Echo-canceling speakerphone	EC_SPEAKERPHONE	A handsfree audio device with echo cancellation capable of fullduplex operation.
Videophone, no echo reduction	VIDEOPHONE	The audio part of a hands-free AV communication device designed for host-based echo cancellation.
Echo-suppressing videophone	ES_VIDEOPHONE	The audio part of a hands-free AV communication device with echo suppression capable of half-duplex operation.
Echo-canceling videophone	EC_VIDEOPHONE	The audio part of a hands-free AV communication device with echo cancellation capable of full-duplex operation.
Phone line	PHONE_LINE	May be an analog telephone line jack, an ISDN line, a proprietary PBX interface, or a wireless link.

Type	Identifier	Description
Telephone	TELEPHONE	Device can be used as a telephone. When not in use as a telephone, handset is used as a bi-directional audio device.
Down Line Phone	DOWN_LINE_PHONE	A standard telephone set connected to the device. When not in use as a telephone, it can be used as a bi-directional audio device.
Analog connection	ANALOG	A generic analog connection.
Digital audio connection	DIGITAL	A generic digital audio connection.
Line connection	LINE	An analog connection at standard line levels. Usually uses 3.5mm connector.
S/PDIF interface	S/PDIF	An S/PDIF digital audio connection.
ADAT Lightpipe	ADAT	An optical connection carrying the Alesis Digital Audio Tape stream. Contact Alesis for technical information.
TDIF	TDIF	Tascam Digital Interface. Contact Tascam for technical information.
MADI	MADI	Multi-channel Audio Digital Interface as defined by AES.
DisplayPort connection	DP	The audio part of a DisplayPort digital connection.
HDMI Input connection	HDMI-IN	The audio part of an HDMI Input digital connection. (Note: HDMI-Out is treated as a specialised AVData Entity)
Phonograph	PHONOGRAPH	Analog vinyl record player.
TV Tuner Audio	TV_TUNER	Audio part of TV tuner.
Satellite Receiver Audio	SAT_TUNER	Audio part of satellite receiver.
Cable Tuner Audio	CABLE_TUNER	Audio part of cable tuner.
DSS Audio	DSS_TUNER	Audio part of DSS receiver.
Radio Receiver	RADIO_TUNER	AM/FM radio receiver.
Level Calibration Noise Source	LEVEL_NOISE	Internal Noise source for level calibration (MPEG decoding, Dolby Prologic™, AC-3 etc.)
Equalization Noise	EQ_NOISE	Internal Noise source for measurements.
Radio Transmitter	RADIO_TX	AM/FM radio transmitter.
TV Transmitter	TV_TX	The Audio part of a TV transmitter
Recorder	RECORDER	A recording system.
CD player	CD	Compact Disc player/recorder.
DAT	DAT	Digital Audio Tape player/recorder.
DCC	DCC	Digital Compact Cassette player/recorder.
Compressed Audio Player	COMPRESSED	Compressed Audio player/recorder.
Analog Tape	ANALOG_TAPE	The audio part of an Analog Video Tape Player/Recorder.
Digital Tape	DIGITAL_TAPE	The audio part of a Digital Video Tape Player/Recorder.
VCR Audio	VCR	The Audio part of a Video Cassette Player/Recorder.
Video Disc Audio	VD	The Audio part of a VideoDisc player/recorder.
DVD Audio	DVD	The Audio part of a DVD player.
BluRay Audio	BLU_RAY	The Audio part of a Blu-Ray player.
Synthesizer	SYNTH	Synthesizer.
Piano	PIANO	Piano.
Guitar	GUITAR	Guitar.
Drums/Rhythm	RHYTHM	Percussion Instrument.
Other Musical Instrument	INSTRUMENT	Undefined Musical Instrument.
Vibro-Kinetic Actuator Inertial	VIBRO_INERTIAL	Vibro-kinetic Actuator of the inertial type.

Type	Identifier	Description
Vibro-Kinetic Actuator Displacement	VIBRO_DISPLACEMENT	Vibro-kinetic Actuator of the displacement type.
Other	OTHER	An audio device other than any of the ones listed above.