

BS EN 62379-2:2009



BSI Standards Publication

Common control interface for networked digital audio and video products —

Part 2: Audio

bsi.

...making excellence a habit.
AVA Documents™
Document Sharing Hub

National foreword

This British Standard is the UK implementation of EN 62379-2:2009. It is identical to IEC 62379-2:2008.

The UK participation in its preparation was entrusted to Technical Committee EPL/100, Audio, video and multimedia systems and equipment.

A list of organizations represented on this committee can be obtained on request to its secretary.

This publication does not purport to include all the necessary provisions of a contract. Users are responsible for its correct application.

© BSI 2010

ISBN 978 0 580 59627 8

ICS 33.160.01

Compliance with a British Standard cannot confer immunity from legal obligations.

This British Standard was published under the authority of the Standards Policy and Strategy Committee on 31 January 2010

Amendments issued since publication

Amd. No.	Date	Text affected
----------	------	---------------

EUROPEAN STANDARD
NORME EUROPÉENNE
EUROPÄISCHE NORM

EN 62379-2

August 2009

ICS 33.160.01

English version

**Common control interface
for networked digital audio and video products -
Part 2: Audio
(IEC 62379-2:2008)**

Interface de commande commun
destiné aux produits audio et vidéo
numériques connectés en réseau -
Partie 2: Audio
(CEI 62379-2:2008)

Gemeinsame Steuerschnittstelle
für netzwerkbetriebene digitale
Audio- und Videogeräte -
Teil 2: Audio
(IEC 62379-2:2008)

This European Standard was approved by CENELEC on 2009-07-01. CENELEC members are bound to comply with the CEN/CENELEC Internal Regulations which stipulate the conditions for giving this European Standard the status of a national standard without any alteration.

Up-to-date lists and bibliographical references concerning such national standards may be obtained on application to the Central Secretariat or to any CENELEC member.

This European Standard exists in three official versions (English, French, German). A version in any other language made by translation under the responsibility of a CENELEC member into its own language and notified to the Central Secretariat has the same status as the official versions.

CENELEC members are the national electrotechnical committees of Austria, Belgium, Bulgaria, Cyprus, the Czech Republic, Denmark, Estonia, Finland, France, Germany, Greece, Hungary, Iceland, Ireland, Italy, Latvia, Lithuania, Luxembourg, Malta, the Netherlands, Norway, Poland, Portugal, Romania, Slovakia, Slovenia, Spain, Sweden, Switzerland and the United Kingdom.

CENELEC

European Committee for Electrotechnical Standardization
Comité Européen de Normalisation Electrotechnique
Europäisches Komitee für Elektrotechnische Normung

Central Secretariat: Avenue Marnix 17, B - 1000 Brussels

© 2009 CENELEC - All rights of exploitation in any form and by any means reserved worldwide for CENELEC members.

Ref. No. EN 62379-2:2009 E

 **Documents**
Document Sharing Hub

Foreword

The text of document 100/1405/FDIS, future edition 1 of IEC 62379-2, prepared by technical area 4, Digital system interfaces and protocols, of IEC TC 100, Audio, video and multimedia systems and equipment, was submitted to the IEC-CENELEC parallel vote and was approved by CENELEC as EN 62379-2 on 2009-07-01.

The following dates were fixed:

- latest date by which the EN has to be implemented
at national level by publication of an identical
national standard or by endorsement (dop) 2010-04-01
- latest date by which the national standards conflicting
with the EN have to be withdrawn (dow) 2012-07-01

Annex ZA has been added by CENELEC.

Endorsement notice

The text of the International Standard IEC 62379-2:2008 was approved by CENELEC as a European Standard without any modification.

Annex ZA (normative)

Normative references to international publications with their corresponding European publications

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

NOTE When an international publication has been modified by common modifications, indicated by (mod), the relevant EN/HD applies.

<u>Publication</u>	<u>Year</u>	<u>Title</u>	<u>EN/HD</u>	<u>Year</u>
IEC 62379-1	2007	Common control interface for networked digital audio and video products - Part 1: General	EN 62379-1	2007
AES3	2003	AES standard for digital audio engineering - Serial transmission format for two-channel linearly represented digital audio data	-	-
AES10	2008	AES recommended practice for digital audio engineering - Serial Multichannel Audio Digital Interface (MADI)	-	-
AES50	2005	AES standard for digital audio engineering - High-resolution multi-channel audio interconnection	-	-
ITU-T Recommendation G.711	⁻¹⁾	Pulse code modulation (PCM) of voice frequencies	-	-
ITU-T Recommendation G.722	⁻¹⁾	7 kHz audio-coding within 64 kbit/s	-	-
ITU-T Recommendation J.41	⁻¹⁾	Characteristics of equipment for the coding of analogue high quality sound programme signals for transmission on 384 kbit/s channels	-	-
ITU-T Recommendation J.57	⁻¹⁾	Transmission of digital studio quality sound signals over H1 channels	-	-

¹⁾ Undated reference.

CONTENTS

INTRODUCTION.....	6
1 Scope.....	7
2 Normative references	7
3 Terms, definitions and abbreviations	7
3.1 Abbreviations	7
4 Audio format definitions.....	8
4.1 Audio signal format definitions.....	8
4.1.1 Audio parameters	8
4.1.2 Audio signal formats	9
4.2 Audio transport format definitions	12
4.3 Audio metadata format definitions	12
5 MIB definitions for audio blocks	13
5.1 General.....	13
5.2 Type definitions.....	13
5.2.1 Textual conventions.....	13
5.2.2 Sequences	15
5.3 Audio port and associated managed object type definitions	17
5.3.1 Generic port functionality.....	17
5.3.2 AES3 ancillary data	18
5.3.3 Phantom power	19
5.3.4 Audio locked to reference	19
5.4 Other audio block and associated managed object type definitions.....	20
5.4.1 Audio mixer blocks	20
5.4.2 Audio crosspoint blocks	23
5.4.3 Audio clip player blocks	26
5.4.4 Audio limiter blocks	29
5.4.5 Audio converter blocks	31
5.4.6 Audio level alarm blocks.....	32
6 Status broadcasts.....	34
6.1 General.....	34
6.2 Type definitions.....	34
6.2.1 Textual conventions.....	34
6.2.2 Sequences	34
6.3 Audio formats mapping.....	34
6.3.1 audioFormatsMapTable	35
6.3.2 audioFormatsMapEntry.....	35
6.3.3 afmNumber.....	35
6.3.4 afmFormat.....	35
6.4 Page formats.....	35
6.4.1 Audio port page	35
6.4.2 AES3 ancillary data page	35
6.4.3 Audio mixer page.....	36
6.4.4 Audio crosspoint page	36
6.4.5 Audio clip player page	37
6.4.6 Audio limiter page.....	37

6.4.7	Audio converter page	38
6.4.8	Audio level alarm page	38
6.5	Page groups	39
6.5.1	audioPorts	39
6.5.2	standardAudioBlocks	39
6.5.3	audioAlarms	40
Annex A (informative)	Machine-readable audio format definitions	41
Annex B (informative)	Machine-readable audio block definitions	56
Annex C (informative)	Machine-readable status page group definitions	74
Annex D (informative)	Machine-readable status page MIB definitions	75
Annex E (informative)	Worked examples	77
Annex F (informative)	Tree of example audio formats	86
Figure 1	– Audio port blocks	17
Figure 2	– Audio mixer block	21
Figure 3	– Audio crosspoint block	23
Figure 4	– Audio clip player block	26
Figure 5	– Audio limiter block	29
Figure 6	– Audio converter block	31
Figure 7	– Audio level alarm block	32
Table 1	– Managed objects for audio ports	17
Table 2	– Managed objects for AES3 ancillary data	18
Table 3	– Managed objects for phantom power	19
Table 4	– Managed objects for audio locked	20
Table 5	– Managed objects for audio mixer blocks	21
Table 6	– Managed objects for audio crosspoint blocks	23
Table 7	– Managed objects for audio clip player blocks	27
Table 8	– Managed objects for audio limiter blocks	30
Table 9	– Managed objects for audio converter blocks	31
Table 10	– Managed objects for audio level alarm blocks	33
Table 11	– Managed objects for audio format mappings	35
Table 12	– Status entries for audio port page	35
Table 13	– Status entries for AES3 ancillary data page	36
Table 14	– Status entries for audio mixer page	36
Table 15	– Status entries for audio crosspoint page	37
Table 16	– Status entries for audio clip player page	37
Table 17	– Status entries for audio limiter page	38
Table 18	– Status entries for audio converter page	38
Table 19	– Status entries for audio level alarm page	39

INTRODUCTION

IEC 62379 specifies the common control interface, a protocol for managing equipment which conveys audio and/or video across digital networks.

This part of IEC 62379 specifies those aspects that are specific to audio equipment.

An introduction to the common control interface is given in IEC 62739-1.

COMMON CONTROL INTERFACE FOR NETWORKED DIGITAL AUDIO AND VIDEO PRODUCTS –

Part 2: Audio

1 Scope

This part of IEC 62379 specifies aspects of the common control interface of IEC 62379-1 that are specific to audio.

2 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

AES3-2003, *AES standard for digital audio — Digital input-output interfacing — Serial transmission format for two-channel linearly represented digital audio data*

AES10-2003, *AES recommended practice for digital audio engineering – Serial multichannel audio digital interface (MADI)*

AES50-2005, *AES standard for digital audio engineering – High-resolution multi-channel audio interconnection (HRMAI)*

IEC 62379-1:2007, *Common control interface for networked audio and video products – Part 1: General*

ITU-T Recommendation G.711, *Pulse code modulation (PCM) of voice frequencies*

ITU-T Recommendation G.722, *7kHz audio-coding within 64 kbit/s*

ITU-T Recommendation J.41, *Characteristics of equipment for the coding of analogue high quality sound programme signals for transmission on 384 kbit/s channels*

ITU-T Recommendation J.57, *Transmission of digital studio quality sound signals over H1 channels*

3 Terms, definitions and abbreviations

For the purposes of this document, the terms and definitions given in IEC 62379-1 apply along with the following abbreviations.

3.1 Abbreviations

3.1.1
pulse code modulation
PCM

3.1.2
motion pictures expert group
MPEG

3.1.3 advanced audio coding AAC

4 Audio format definitions

4.1 Audio signal format definitions

At any point in the audio signal chain, the audio data will be in a particular format. For management purposes, the format shall be identified by an object identifier, either a “common control interface standard” object identifier defined in this standard or an object identifier defined elsewhere.

NOTE Permitting audio format identifiers to be defined outside this standard allows use of proprietary formats within the standard protocol and also allows industry standard formats to emerge that may eventually be incorporated into future revisions of this standard.

4.1.1 Audio parameters

4.1.1.1 General

The definitions in 4.1.2 make reference to parameters which provide additional information about the format. These parameters shall be mapped to sub-identifier values as specified in the other subclauses of 4.1.1. Any parameter may be unspecified.

The sub-identifier values shall be appended to the object identifiers as additional arcs, in the order in which the parameters are listed in the relevant subclause of 4.1.2; except that if a parameter is unspecified and either it is the last parameter or all subsequent parameters are also unspecified, then it shall be omitted.

NOTE For all parameters, “unspecified” is coded as zero, so this rule ensures that the OID does not end with a zero arc.

EXAMPLE: If the last two parameters are bit depth and sampling frequency, then 16-bit 48kHz would be coded as .16.48000, 16-bit (with sampling frequency unspecified) as .16, and 48kHz (with bit depth unspecified) as .0.48000.

4.1.1.2 Channel arrangement

The sub-identifier for channel arrangement shall be a value of the following type:

```
ChannelArrangement ::= INTEGER {
    unspecified          (0),
    discreteMono         (1),
    stereo               (2),
    jointStereo          (3),
    surround             (4),
    surroundWithDownmix (5)
} (unspecified.. surroundWithDownmix)
```

`discreteMono` shall indicate that each channel (if there is more than one) is a separate audio signal.

`stereo` should only be used with an even number of channels, and shall indicate that each pair of channels (if there is more than one pair) is a separate stereo audio signal, with the first channel of the pair being the left channel.

`jointStereo` should only be used with an even number of channels, and shall indicate that each pair of channels (if there is more than one pair) is a separate stereo audio signal, with the first channel of the pair being the M channel and the second the S channel.

`surround` and `surroundWithDownmix` should only be used with 3 to 8 channels (inclusive). Each shall indicate that the first six channels contain left, right, centre, low frequency effects, left surround, and right surround respectively. `surround` shall indicate that the next two channels contain left and right channels (respectively) of a stereo signal suitable for matrix decoding. `surroundWithDownmix` shall indicate that the next two channels contain left and right channels (respectively) of a stereo downmix.

Where more than one arrangement can be used to describe a format, the smallest applicable value should be used.

EXAMPLE 1 A single stereo pair (2 channels) could be described by values `stereo` (2), `surround` (4), or `surroundWithDownmix` (5). The smallest of these values, i.e. 2, should be used.

EXAMPLE 2 Surround sound with no accompanying stereo signal (6 channels) could be described by values `surround` (4), or `surroundWithDownmix` (5). The smaller of these values, i.e. 4, should be used.

4.1.1.3 Number of channels

The sub-identifier for number of channels shall be a value of the following type:

```
NumberChannels ::= INTEGER
-- An integer representing the number of audio channels
-- A value of zero shall indicate unspecified.
```

4.1.1.4 Bit depth

The sub-identifier for bit depth shall be a value of the following type:

```
BitDepth ::= INTEGER
-- An integer representing the audio bit depth in bits per sample.
-- A value of zero shall indicate unspecified.
```

4.1.1.5 Sampling frequency

The sub-identifier for sampling frequency shall be a value of the following type:

```
SamplingFrequency ::= INTEGER
-- An integer representing the audio sampling frequency in Hz
-- A value of zero shall indicate unspecified.
```

4.1.1.6 Bit rate

The sub-identifier for bit rate shall be a value of the following type:

```
BitRate ::= INTEGER
-- An integer representing the bit rate of the encoded signal in bits per
-- second.
-- A value of zero shall indicate unspecified.
```

4.1.2 Audio signal formats

Audio signal formats shall be rooted at the following location in the MIB tree:

```
iec62379          OBJECT IDENTIFIER ::= { iso(1) standard(0) 62379 }
audioFormat       OBJECT IDENTIFIER ::= { iec62379 audio(2) format(2) }
audioSignalFormat OBJECT IDENTIFIER ::= { audioFormat signal(1) }
```

The following definitions shall be used to identify the specified formats.

NOTE Annex A contains an exemplar set of formats defined by this standard.

4.1.2.1 Unspecified audio

```
unspecifiedAudio      OBJECT IDENTIFIER ::=
                        { audioSignalFormat unspecified(0) }
-- wildcard - any supported format allowed
```

4.1.2.2 No audio

```
noAudio              OBJECT IDENTIFIER ::= { audioSignalFormat none(1) }
-- indicates the output is muted
```

4.1.2.3 Analogue audio

```
analogueAudio        OBJECT IDENTIFIER ::= { audioSignalFormat analogue(2)
}
-- analogue audio
```

The analogue audio identifier shall have two parameters. The first shall be the channel arrangement and the second shall be the number of channels.

4.1.2.4 PCM audio

```
pcmAudio             OBJECT IDENTIFIER ::= { audioSignalFormat pcm(3) }
-- linear PCM audio
```

The linear PCM format identifier shall have four parameters. The first shall be the channel arrangement, the second shall be the number of channels, the third shall be the bit depth and the fourth shall be the sampling frequency.

4.1.2.5 MPEG-1 layer 2 audio

```
mp2Audio             OBJECT IDENTIFIER ::= { audioSignalFormat mp2(4) }
-- MPEG-1 layer 2 audio
```

The MP2 format identifier shall have four parameters. The first shall be the channel arrangement, the second shall be the number of channels, the third shall be the sampling frequency, and the fourth shall be the bit rate.

4.1.2.6 MPEG-1 layer 3 audio

```
mp3Audio             OBJECT IDENTIFIER ::= { audioSignalFormat mp3(5) }
-- MPEG-1 layer 3 audio
```

The MP3 format identifier shall have four parameters. The first shall be the channel arrangement, the second shall be the number of channels, the third shall be the sampling frequency and the fourth shall be the bit rate.

4.1.2.7 AAC audio

```
aacAudio             OBJECT IDENTIFIER ::= { audioSignalFormat aac(6) }
-- AAC audio
```

```
aacLC                OBJECT IDENTIFIER ::= { aacAudio aacLC(1) }
-- AAC audio with the low complexity profile
```

```
aacMain              OBJECT IDENTIFIER ::= { aacAudio aacMain(2) }
-- AAC audio with the main profile
```

```
aacSRS               OBJECT IDENTIFIER ::= { aacAudio aacSRS(3) }
-- AAC audio with the sample-rate-scalable profile
```

```

aacLTP                OBJECT IDENTIFIER ::= { aacAudio aacLTP(4) }
-- AAC audio with the long term prediction profile

aacLD                 OBJECT IDENTIFIER ::= { aacAudio aacLD(5) }
-- AAC audio with the low delay profile

```

The AAC format identifier shall have four parameters. The first shall be the channel arrangement, the second shall be the number of channels, the third shall be the sampling frequency and the fourth shall be the bit rate.

4.1.2.8 Audio conforming to ITU-T Recommendation G.711

```

g711Audio             OBJECT IDENTIFIER ::= { audioSignalFormat g711(7) }
-- G711 audio

g711ALaw              OBJECT IDENTIFIER ::= { g711 aLaw(1) }
-- G711 audio, A-Law encoded

g711MuLaw             OBJECT IDENTIFIER ::= { g711 muLaw(2) }
-- G711 audio, mu-Law encoded

```

4.1.2.9 Audio conforming to ITU-T Recommendation G.722

```

g722Audio             OBJECT IDENTIFIER ::= { audioSignalFormat g722(8) }
-- G722 audio

```

The G722 identifier shall have one parameter. This shall be the bit rate.

4.1.2.10 APT-X audio

```

aptXAudio             OBJECT IDENTIFIER ::= { audioSignalFormat aptX(9) }
-- apt-X audio

```

The APT-X format identifier shall have five parameters. The first shall be the channel arrangement, the second shall be the number of channels, the third shall be the bit depth, the fourth shall be the sampling frequency, and the fifth shall be the bit rate.

4.1.2.11 Enhanced APT-X audio

```

enhancedAptXAudio     OBJECT IDENTIFIER ::= { audioSignalFormat enhAptX(10) }
-- enhanced apt-X audio

```

The enhanced APT-X format identifier shall have five parameters. The first shall be the channel arrangement, the second shall be the number of channels, the third shall be the bit depth, the fourth shall be the sampling frequency and the fifth shall be the bit rate.

4.1.2.12 Audio conforming to ITU-T Recommendation J.41

```

j41Audio              OBJECT IDENTIFIER ::= { audioSignalFormat j41(11) }
-- J41 audio

j41ALawA              OBJECT IDENTIFIER ::= { j41 aLawA(1) }
-- J41 audio using A-law companding, variant A

j41ALawB              OBJECT IDENTIFIER ::= { j41 aLawB(2) }
-- J41 audio using A-law companding, variant B

j41Nic                OBJECT IDENTIFIER ::= { j41 nic(3) }
-- J41 audio using near instantaneous companding

```

4.1.2.13 Audio conforming to ITU-T Recommendation J.57

```
j57Audio          OBJECT IDENTIFIER ::= { audioSignalFormat j57(12) }
-- J57 audio

j57H11            OBJECT IDENTIFIER ::= { j57 h11(1) }
-- J57 audio using an H11 channel

j57H12            OBJECT IDENTIFIER ::= { j57 h12(2) }
-- J57 audio using an H12 channel
```

4.1.2.14 Invalid audio

```
invalidAudio      OBJECT IDENTIFIER ::= { audioSignalFormat invalid(13)
}
-- indicates an error, such as inability to decode a signal earlier in
-- the chain
```

4.2 Audio transport format definitions

Audio transport formats shall be rooted at the following location in the MIB tree:

```
audioTransportFormat OBJECT IDENTIFIER ::= { audioFormat transport(2) }
```

The following definitions shall be used to identify the specified transport formats.

```
unspecifiedTransport OBJECT IDENTIFIER ::=
{ audioTransportFormat unspecified(0) }

analogue      OBJECT IDENTIFIER ::= { audioTransportFormat analogue(1) }
AES3          OBJECT IDENTIFIER ::= { audioTransportFormat aes3(2) }
AES10         OBJECT IDENTIFIER ::= { audioTransportFormat aes10(3) }
AES50         OBJECT IDENTIFIER ::= { audioTransportFormat aes50(4) }
```

4.3 Audio metadata format definitions

Audio metadata formats shall be rooted at the following location in the MIB tree:

```
audioMetadataFormat OBJECT IDENTIFIER ::= { audioFormat metadata(3) }
```

The following definitions shall be used to identify the specified metadata formats.

```
unspecifiedMetadata OBJECT IDENTIFIER ::=
{ audioMetadataFormat unspecified(0) }
```

5 MIB definitions for audio blocks

5.1 General

This clause defines a set of managed object types for representing control functions in network controlled audio equipment. The format of the definitions is as specified in IEC 62379-1.

For management purposes, a piece of audio equipment shall be modelled as a number of discrete audio blocks and audio connectors, as specified in IEC 62379-1. Each audio block may have zero or more inputs and zero or more outputs, and each input or output may carry one or more channels. Each audio connector shall connect one audio block output to one audio block input with a one-to-one mapping of channels between the blocks.

NOTE 1 A piece of equipment may be fixed-function, in which case the number of audio blocks present and the connections between them will be immutable, or it may be programmable, in which case the number of audio blocks present and/or the connections between them may be changed by the user.

Each audio block shall be modelled either by one of the standard audio block types defined in this standard or by an audio block type defined elsewhere. Associated with each defined block type shall be a (possibly empty) group of managed object types that represent the control functions for that block. A block type shall be identified by the node in the object identifier tree that is the root node for the group of managed object types associated with that block type.

NOTE 2 Permitting audio block types to be defined outside this standard allows control of proprietary functions using the standard protocol and also allows industry standard block types to emerge that may eventually be incorporated into future revisions of this standard.

NOTE 3 An empty group of managed object types is permitted to allow for blocks that have no associated control functions.

NOTE 4 Annex E contains worked examples of the block structure.

5.2 Type definitions

In addition to the types defined in IEC 62379-1, the following types are used to specify the syntax of the abstract data structures representing managed object values.

5.2.1 Textual conventions

```
AudioTransportType ::= OBJECT IDENTIFIER
-- A reference to the transport used for an audio connection.
-- The value may be defined in 4.2, or in a sub-part of IEC 62379-5, or
-- elsewhere.
```

```
AudioLevel ::= INTEGER {
    mInfinity (-20000),
    fullScale (0),
    pInfinity (20000)
} (mInfinity..pInfinity)
-- An absolute or relative audio level in units of 0.01dB.
```

```
AudioPhase ::= INTEGER (-18000..18000)
-- An absolute or relative phase value in units of 0.01 degree.
```

```
AudioQuality ::= INTEGER {
    worst (1),
    low (32),
    high (96),
    best (127)
} (worst..best)
-- An enumeration identifying an audio signal processing quality level.
```

```
-- Semantics are equipment specific. Equipment that supports selectable
-- quality levels shall as a minimum support the values low and high.
```

```
AudioChannel ::= INTEGER {
    left  (1),
    right (2)
} (1..240)
-- An enumeration identifying an audio channel. Stereo equipment shall
-- accept the values left and right. Multi-channel equipment should
-- accept a contiguous range of values starting from 1 and may disregard
-- the association of the values 1 and 2 with left and right.
```

```
Aes3ChannelData ::= OCTET STRING (SIZE(24))
-- The channel status data associated with an AES3 channel.
```

```
Aes3UserData ::= OCTET STRING (SIZE(24))
-- User data associated with an AES3 channel.
```

```
AudioClipSelector ::= INTEGER (1..2147483647)
-- An enumeration identifying an audio clip. Semantics are equipment
-- specific.
```

```
AudioClipActivation ::= INTEGER {
    play      (1),
    stop      (2),
    stopAtEnd (3)
} (play..stopAtEnd)
-- An enumeration for audio clip playback control.
```

```
AudioClipTrackControl ::= INTEGER {
    previous (1),
    next     (2)
} (previous..next)
-- An enumeration for audio clip track control.
```

```
AudioClipRepeat ::= INTEGER {
    none  (1),
    track (2),
    all   (3)
} (none..all)
-- An enumeration for audio clip repeat control.
```

```
AudioRecoveryMode ::= INTEGER {
    auto (1),
    slow (2),
    fast (3)
} (auto..fast)
-- An enumeration identifying an audio limiter recovery mode. Semantics
-- are equipment specific.
```

```
AudioAlarmStatus ::= INTEGER {
    ok      (1),
    warning (2),
    failure (3)
} (ok..failure)
-- An enumeration representing the status of an audio alarm
```

```
AudioLevelAlarmType ::= INTEGER {
    lower (1),
    higher (2)
} (lower..higher)
-- An enumeration representing the trigger condition for an audio level
```

```
-- alarm. If lower, the alarm is triggered by the audio level being less
-- than the threshold; if higher, the alarm is triggered by the audio
-- level being greater than the threshold.
```

5.2.2 Sequences

```
APortEntry ::= SEQUENCE {
    aPortBlockId      BlockId,
    aPortDirection    PortDirection,
    aPortFormat        MediaFormat,
    aPortTransport     AudioTransportType,
    aPortName          Utf8String
}

Aes3DataEntry ::= SEQUENCE {
    aes3BlockId        BlockId,
    aes3ChannelId       AudioChannel,
    aes3ChannelData     Aes3ChannelData,
    aes3UserData        Aes3UserData,
    aes3ValidityError   TruthValue
}

APhantomEntry ::= SEQUENCE {
    aPhantomBlockId    BlockId,
    aPhantomEnabled     TruthValue,
    aPhantomLevel       CardinalNumber
}

ALockedEntry ::= SEQUENCE {
    aLockedBlockId      BlockId,
    aLockedTime          CardinalNumber,
    aLockedSamplesInserted CardinalNumber,
    aLockedSamplesDropped CardinalNumber
}

AMixerBlockEntry ::= SEQUENCE {
    aMixerBlockId        BlockId,
    aMixerFadeDuration    CardinalNumber,
    aMixerFadeNow         TruthValue
}

AMixerInputEntry ::= SEQUENCE {
    aMixerInputBlockId    BlockId,
    aMixerInputNumber      IndexNumber,
    aMixerInputLevel       AudioLevel,
    aMixerInputFadeToLevel AudioLevel,
    aMixerInputDelay        CardinalNumber
}

ACrosspointBlockEntry ::= SEQUENCE {
    aCrosspointBlockId    BlockId,
    aCrosspointConfigure   TruthValue,
    aCrosspointCopy        BlockId
}

ACrosspointPathEntry ::= SEQUENCE {
    aCrosspointPathBlockId BlockId,
    aCrosspointPathSrc      AudioChannel,
    aCrosspointPathDst      AudioChannel,
    aCrosspointPathGain     AudioLevel,
    aCrosspointPathNewGain  AudioLevel,
    aCrosspointPathPhase    AudioPhase,
    aCrosspointPathNewPhase AudioPhase
}
```

```

}
```

```

AClipPlayerBlockEntry ::= SEQUENCE {
    acpBlockId          BlockId,
    acpSelection         AudioClipSelector,
    acpMainFaderLevelNow AudioLevel,
    acpMainFaderLevelStart AudioLevel,
    acpMainFaderLevelStop AudioLevel,
    acpMainFaderDuration CardinalNumber,
    acpClipFaderLevelNow AudioLevel,
    acpClipFaderLevelStart AudioLevel,
    acpClipFaderLevelStop AudioLevel,
    acpClipFaderDuration CardinalNumber,
    acpStartDelay        IntegerNumber,
    acpActivation         AudioClipActivation,
    acpTrackControl       AudioClipTrackControl,
    acpTrackNumber        CardinalNumber,
    acpRepeat             AudioClipRepeat
}
```

```

AClipSequenceEntry ::= SEQUENCE {
    aClipSequenceBlockId BlockId,
    aClipSequenceId       AudioClipSelector,
    aClipSequenceName      Utf8String
}
```

```

ALimiterBlockEntry ::= SEQUENCE {
    alimiterBlockId      BlockId,
    alimiterThreshold     AudioLevel,
    alimiterAttackTime    CardinalNumber,
    alimiterGainMakeup    AudioLevel,
    alimiterRecoveryTime  CardinalNumber,
    alimiterRecoveryMode  AudioRecoveryMode
}
```

```

AConverterBlockEntry ::= SEQUENCE {
    aConverterBlockId      BlockId,
    aConverterQuality       AudioQuality,
    aConverterEnabled       TruthValue,
    aConverterDithering     TruthValue,
    aConverterOutputFormat  MediaFormat,
    aConverterError         TruthValue
}
```

```

ALevelAlarmBlockEntry ::= SEQUENCE {
    alaBlockId            BlockId,
    alaType               AudioLevelAlarmType,
    alaThreshold           AudioLevel,
    alaWarningTime         CardinalNumber,
    alaFailureTime         CardinalNumber,
    alaCounter             CardinalNumber,
    alaEnabled             TruthValue,
    alaStatus             AudioAlarmStatus
}
```

5.3 Audio port and associated managed object type definitions

5.3.1 Generic port functionality

All audio inputs to and outputs from the unit shall be represented using an audio port block. A base table of managed objects provides control common to all ports; extension tables provide functionality specific to certain types of port.

An audio port block shall have the following structure, where *c* is the number of channels on the input or output:

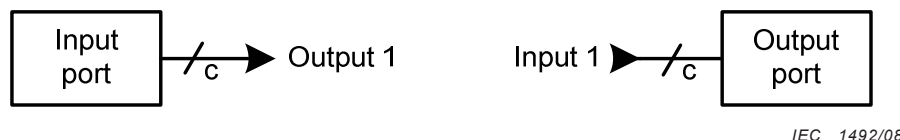


Figure 1 – Audio port blocks

The group of objects in Table 1 shall be implemented by all compliant audio equipment that contains one or more audio ports. The root node for these objects shall be

```
{ iso(1) standard(0) iec62379 audio(2) audioMIB(1) audioPort(1) }
```

This node shall be used as the audio block type identifier for audio port blocks.

Table 1 – Managed objects for audio ports

Identifier	Syntax	Index	Readable	Writable	Volatile	Status
aPortTable(1)	SEQUENCE OF APortEntry		none	none	no	m
lPortEntry(1)	APortEntry		none	none	no	m
lPortBlockId(1)	BlockId	yes	none	none	no	m
lPortDirection(2)	PortDirection		listener	none	no	m
lPortFormat(3)	MediaFormat		listener	none	yes	m
lPortTransport(4)	AudioTransportType		listener	none	no	o
lPortName(5)	Utf8String		listener	supervisor	no	o

5.3.1.1 aPortTable

A table of audio port descriptors for this unit. Each audio port on the unit has a corresponding entry in this table.

5.3.1.2 aPortEntry

An entry in the audio port table.

5.3.1.3 aPortBlockId

The block identifier for this port. Used as an index when accessing the audio port table.

5.3.1.4 aPortDirection

The direction (input or output) of this port.

5.3.1.5 aPortFormat

The format of the audio data currently being received or transmitted via this port. If the port is not active, the value `noAudio` shall be returned.

5.3.1.6 aPortTransport

The type of transport used by the port.

5.3.1.7 aPortName

The name assigned to this port. This is an arbitrary text string assigned by the system manager.

5.3.2 AES3 ancillary data

The group of objects in Table 2 shall be implemented by all compliant audio equipment that allows access to AES3 ancillary data. The root node for these objects shall be

```
{ iso(1) standard(0) iec62379 audio(2) audioMIB(1) audioPort(1) }
```

Table 2 – Managed objects for AES3 ancillary data

Identifier	Syntax	Index	Readable	Writable	Volatile	Status
aes3DataTable(2)	SEQUENCE OF Aes3DataEntry		none	none	no	m
└aes3DataEntry(1)	Aes3DataEntry		none	none	no	m
└└aes3BlockId(1)	BlockId	yes	none	none	no	m
└└aes3ChannelId(2)	AudioChannel	yes	none	none	no	m
└└└aes3ChannelData(3)	Aes3ChannelData		listener	none	yes	o
└└└aes3UserData(4)	Aes3UserData		listener	none	yes	o
└└└└aes3ValidityError(5)	TruthValue		listener	supervisor	yes	o

5.3.2.1 aes3DataTable

A table of AES3 ancillary data descriptors for this unit. Each channel of each AES3 port on the unit has a corresponding entry in this table.

5.3.2.2 aes3DataEntry

An entry in the AES3 ancillary data table.

5.3.2.3 aes3BlockId

The block identifier of the associated audio port. Used as an index when accessing the AES3 ancillary data table.

5.3.2.4 aes3ChannelId

The channel number. Used as an index when accessing the AES3 ancillary data table.

5.3.2.5 aes3ChannelData

The current channel status data (24 bytes) for this channel.

5.3.2.6 aes3UserData

The current user data for this channel.

5.3.2.7 aes3ValidityError

An indication that shows whether this channel's validity bit has been set in any subframe. It shall be automatically set to `true` when a subframe with `V=1` is detected. It shall only be set to `false` as a result of a `SET` operation on this object.

NOTE V=1 indicates an error in the case of linear PCM; for other formats V=1 should be set in every subframe.

5.3.3 Phantom power

The group of objects in Table 3 shall be implemented by all compliant audio equipment that provides phantom power on an audio port. The root node for these objects shall be

```
{ iso(1) standard(0) iec62379 audio(2) audioMIB(1) audioPort(1) }
```

Table 3 – Managed objects for phantom power

Identifier	Syntax	Index	Readable	Writable	Volatile	Status
aPhantomTable(3)	SEQUENCE OF APhantomEntry		none	none	no	m
lAPhantomEntry(1)	APhantomEntry		none	none	no	m
lAPhantomBlockId(1)	BlockId	yes	none	none	no	m
lAPhantomEnabled(2)	TruthValue		listener	operator	no	m
lAPhantomLevel(3)	CardinalNumber		listener	supervisor	no	o

5.3.3.1 aPhantomTable

A table of audio port phantom power descriptors for this unit. Each audio port on the unit that supports phantom power has a corresponding entry in this table.

5.3.3.2 aPhantomEntry

An entry in the phantom power table.

5.3.3.3 aPhantomBlockId

The block identifier of the associated audio port. Used as an index when accessing the phantom power table.

5.3.3.4 aPhantomEnabled

When set to `true`, phantom power is provided by this port. When set to `false`, phantom power is not provided by this port.

5.3.3.5 aPhantomLevel

The level of the phantom power provided in millivolts.

5.3.4 Audio locked to reference

The group of objects in Table 4 shall be implemented by all compliant audio equipment that provides statistics for audio signals being locked to a reference signal. The root node for these objects shall be

```
{ iso(1) standard(0) iec62379 audio(2) audioMIB(1) audioPort(1) }
```

Table 4 – Managed objects for audio locked

Identifier	Syntax	Index	Readable	Writable	Volatile	Status
aLockedTable(3)	SEQUENCE OF ALockedEntry		none	none	no	m
lALockedEntry(1)	ALockedEntry		none	none	no	m
lALockedBlockId(1)	BlockId	yes	none	none	no	m
lALockedTime(2)	CardinalNumber		listener	none	yes	m
lALockedSamplesInserted(3)	CardinalNumber		listener	supervisor	yes	m
lALockedSamplesDropped(4)	CardinalNumber		listener	supervisor	yes	m

5.3.4.1 aLockedTable

A table of audio port locked descriptors for this unit. Each audio port on the unit that supports measurement of lock against the reference has a corresponding entry in this table.

5.3.4.2 aLockedEntry

An entry in the locked table.

5.3.4.3 aLockedBlockId

The block identifier of the associated audio port. Used as an index when accessing the locked table.

5.3.4.4 aLockedTime

The time (in seconds) that the audio signal on this port has been locked to the unit's reference clock.

5.3.4.5 aLockedSamplesInserted

The number of audio samples that have been inserted due to buffer underrun or overrun on this port. The value shall be updated every 10 s by adding the number of samples inserted in the last 10 s to half the previous value.

5.3.4.6 aLockedSamplesDropped

The number of audio samples that have been dropped due to buffer underrun or overrun on this port. The value shall be updated every 10 s by adding the number of samples inserted in the last 10 s to half the previous value.

5.4 Other audio block and associated managed object type definitions**5.4.1 Audio mixer blocks**

An audio mixer block shall have the following structure:

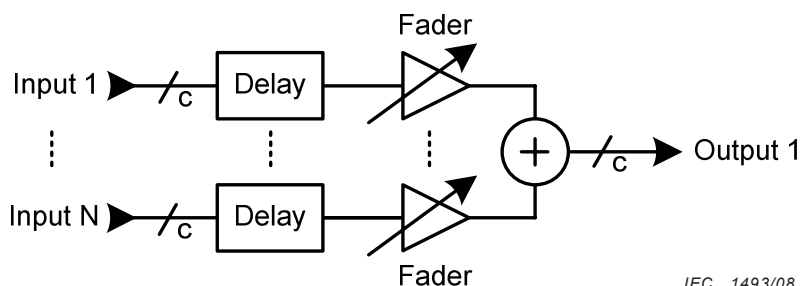


Figure 2 – Audio mixer block

where c is the number of channels on a connection.

NOTE 1 An audio mixer block may be used to represent a simple switched selector or combiner, by limiting the permitted values for the fader level controls to `mInfinity` or `fullScale`.

NOTE 2 The delay function permits audio streams that have passed through various processing or transport paths to be brought back into time alignment, either with other audio streams or with associated video streams. Equipment that does not support this functionality is represented as having a fixed zero delay.

The group of objects in Table 5 shall be implemented by all compliant audio equipment that has a management model that incorporates one or more audio mixer blocks. The root node for these objects shall be

{ iso(1) standard(0) iec62379 audio(2) audioMIB(1) audioMixer(2) }

This node shall be used as the block type identifier for audio mixer blocks.

Table 5 – Managed objects for audio mixer blocks

Identifier	Syntax	Index	Readable	Writable	Volatile	Status
<code>aMixerBlockTable(1)</code>	SEQUENCE OF <code>AMixerBlockEntry</code>		none	none	no	m
<code>lAMixerBlockEntry(1)</code>	<code>AMixerBlockEntry</code>		none	none	no	m
<code>lAMixerBlockId(1)</code>	<code>BlockId</code>	yes	none	none	no	m
<code>lAMixerFadeDuration(2)</code>	<code>CardinalNumber</code>		listener	operator	no	o
<code>lAMixerFadeNow(3)</code>	<code>TruthValue</code>		listener	operator	yes	o
<code>aMixerInputTable(2)</code>	SEQUENCE OF <code>AMixerInputEntry</code>		none	none	no	m
<code>lAMixerInputEntry(1)</code>	<code>AMixerInputEntry</code>		none	none	no	m
<code>lAMixerInputBlockId(1)</code>	<code>BlockId</code>	yes	none	none	no	m
<code>lAMixerInputNumber(2)</code>	<code>IndexNumber</code>	yes	none	none	no	m
<code>lAMixerInputLevel(3)</code>	<code>AudioLevel</code>		listener	operator	no	m
<code>lAMixerInputFadeToLevel(4)</code>	<code>AudioLevel</code>		listener	operator	no	o
<code>lAMixerInputDelay(5)</code>	<code>CardinalNumber</code>		listener	operator	no	o

5.4.1.1 aMixerBlockTable

A table of audio mixer block descriptors for this unit. Each audio mixer block in the unit has a corresponding entry in this table.

5.4.1.2 aMixerBlockEntry

An entry in the audio mixer block table.

5.4.1.3 **aMixerBlockId**

The block identifier for this block. Used as an index when accessing the audio mixer block table.

5.4.1.4 **aMixerFadeDuration**

The transition time (in milliseconds) for a smooth fade or cross-fade operation performed by this block. A value of zero indicates a hard switchover.

5.4.1.5 **aMixerFadeNow**

When set to `true`, causes the block to perform a smooth fade or cross-fade operation. Using values taken from this block's entries in the mixer input table, the unit shall simultaneously ramp the fader level for each input in `aMixerInputEntry` from its current value to the value specified by `aMixerInputFadeToLevel`. Automatically reset to `false` when the operation is complete.

5.4.1.6 **aMixerInputTable**

A table of audio mixer input descriptors for this unit. Each input of each audio mixer block in the unit has a corresponding entry in this table.

5.4.1.7 **aMixerInputEntry**

An entry in the audio mixer input table.

5.4.1.8 **aMixerInputBlockId**

The block identifier of the associated block. Used as an index when accessing the audio mixer input table.

5.4.1.9 **aMixerInputNumber**

The block input number for this input. Used as an index when accessing the audio mixer input table.

5.4.1.10 **aMixerInputLevel**

The fader level for this input. When a level is set, the fader immediately changes to the level specified, over the duration in `aMixerFadeDuration`. For blocks that only support switching between inputs, the only permitted values are `mInfinity` and `fullScale`. Blocks that automatically switch between inputs may reject `SET` operations on this object.

5.4.1.11 **aMixerInputFadeToLevel**

The fader level for this input that will be applied when `aMixerFadeNow` is set to `true`. For blocks that only support switching between inputs, the only permitted values are `mInfinity` and `fullScale`. Blocks that automatically switch between inputs may reject `SET` operations on this object.

5.4.1.12 **aMixerInputDelay**

The delay (in microseconds) applied to samples arriving at this input.

5.4.2 Audio crosspoint blocks

An audio crosspoint block shall have the following structure:

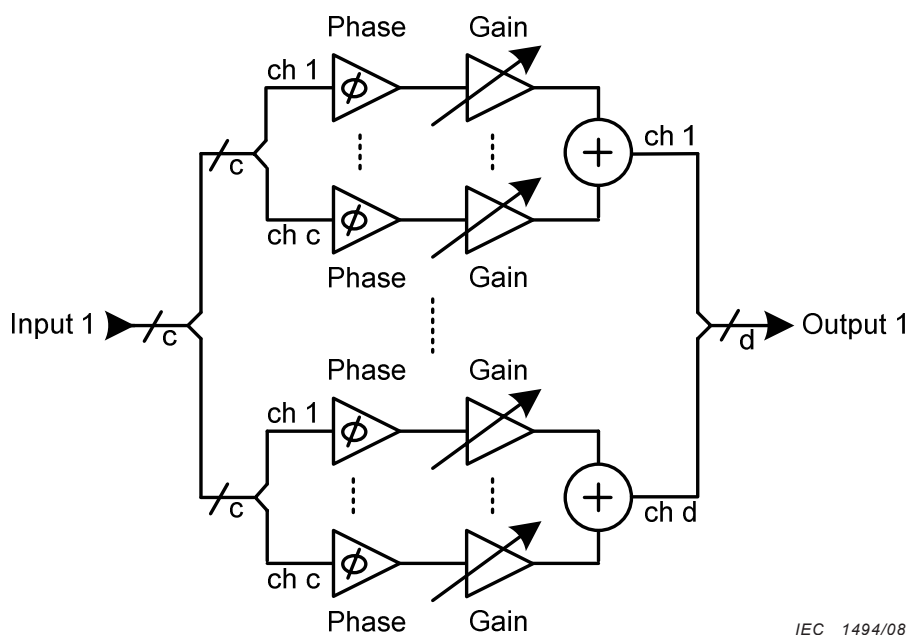


Figure 3 – Audio crosspoint block

where c is the number of input channels and d is the number of output channels.

The group of objects in Table 6 shall be implemented by all compliant audio equipment that has a management model that incorporates one or more audio crosspoint blocks. The root node for these objects shall be

{ iso(1) standard(0) iec62379 audio(2) audioMIB(1) audioCrosspoint(3) }

This node shall be used as the block type identifier for audio crosspoint blocks.

Table 6 – Managed objects for audio crosspoint blocks

Identifier	Syntax	Index	Readable	Writable	Volatile	Status
aCrosspointBlockTable(1)	SEQUENCE OF ACrosspointBlockEntry		none	none	no	m
lACrosspointBlockEntry(1)	ACrosspointBlockEntry		none	none	no	m
lACrosspointBlockId(1)	BlockId	yes	none	none	no	m
lACrosspointConfigure(2)	TruthValue		listener	operator	yes	m
lACrosspointCopy(3)	BlockId		none	operator	yes	o
aCrosspointPathTable(2)	SEQUENCE OF ACrosspointPathEntry		none	none	no	m
lACrosspointPathEntry(1)	ACrosspointPathEntry		none	none	no	m
lACrosspointPathBlockId(1)	BlockId	yes	none	none	no	m
lACrosspointPathSrc(2)	AudioChannel	yes	none	none	no	m
lACrosspointPathDst(3)	AudioChannel	yes	none	none	no	m
lACrosspointPathGain(4)	AudioLevel		listener	operator	no	m
lACrosspointPathNewGain(5)	AudioLevel		listener	operator	no	o
lACrosspointPathPhase(6)	AudioPhase		listener	operator	no	m
lACrosspointPathNewPhase(7)	AudioPhase		listener	operator	no	o

5.4.2.1 **aCrosspointBlockTable**

A table of audio crosspoint block descriptors for this unit. Each audio crosspoint block in the unit has a corresponding entry in this table.

5.4.2.2 **aCrosspointBlockEntry**

An entry in the audio crosspoint block table.

5.4.2.3 **aCrosspointBlockId**

The block identifier for this block. Used as an index when accessing the audio crosspoint block table.

5.4.2.4 **aCrosspointConfigure**

When set to `true`, causes this block to be configured as specified by the corresponding `aCrosspointPathNewGain` and `aCrosspointPathNewPhase` entries in the crosspoint path table. Automatically reset to `false` if the block supports delayed configuration and any of the corresponding entries in the crosspoint path table are modified. Permanently `true` if the block does not support delayed configuration.

5.4.2.5 **aCrosspointCopy**

When set to a block identifier that identifies another audio crosspoint block in the unit with an identical structure to this block, copies the values of `aCrosspointPathGain` and `aCrosspointPathPhase` for each path in the crosspoint from the identified block to this block. If set to a block identifier that does not identify an audio crosspoint block in the unit with an identical structure to this block, the `SET` operation shall be rejected.

NOTE A possible application is the ability to have some common configurations available as presets by creating 'dummy' crosspoint blocks with the required settings which are referenced in the block table but which are not actually part of the audio path.

5.4.2.6 **aCrosspointPathTable**

A table of audio crosspoint path descriptors for this unit. Each path in each audio crosspoint block in the unit has a corresponding entry in this table.

5.4.2.7 **aCrosspointPathEntry**

An entry in the audio crosspoint path table.

5.4.2.8 **aCrosspointPathBlockId**

The block identifier of the associated block. Used as an index when accessing the audio crosspoint path table.

5.4.2.9 **aCrosspointPathSrc**

The source audio channel associated with this path. Used as an index when accessing the audio crosspoint path table.

5.4.2.10 **aCrosspointPathDst**

The destination audio channel associated with this path. Used as an index when accessing the audio crosspoint path table.

5.4.2.11 aCrosspointPathGain

The signal gain from source to destination of this path. If this value is set, the change occurs immediately.

5.4.2.12 aCrosspointPathNewGain

The signal gain from source to destination of this path that will be applied when `aCrosspointConfigure` is set to `true`.

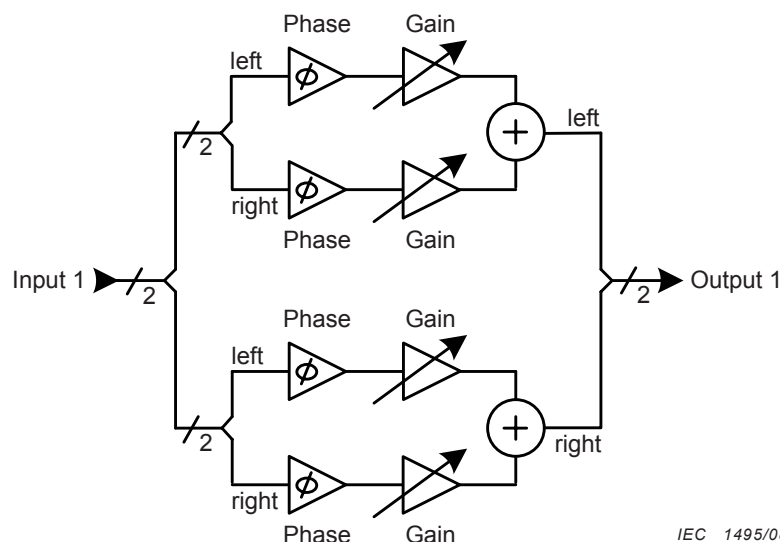
5.4.2.13 aCrosspointPathPhase

The phase shift from source to destination of this path. If this value is set, the change occurs immediately.

5.4.2.14 aCrosspointPathNewPhase

The phase shift from source to destination of this path that will be applied when `aCrosspointConfigure` is set to `true`.

EXAMPLE: The crosspoint block permits a variety of audio functions to be modelled. As an example, consider the following stereo crosspoint:



There are four routes through the crosspoint representing what effect an input channel (the source, represented by `aCrosspointPathSrc`) has on an output channel (the destination, represented by `aCrosspointPathDst`). The source / destination pairs represent:

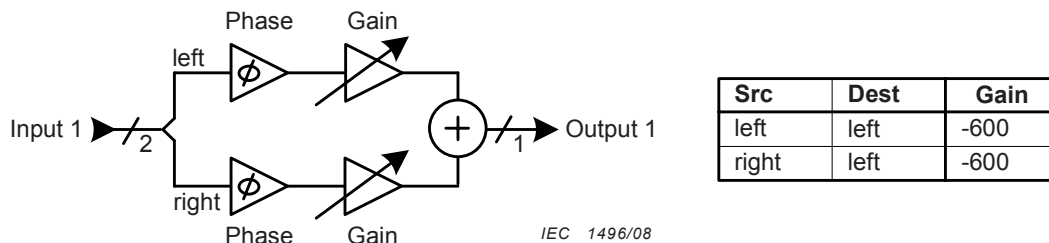
- the effect the left channel of the input has on the left channel of the output;
- the effect the right channel of the input has on the left channel of the output;
- the effect the left channel of the input has on the right channel of the output;
- the effect the right channel of the input has on the right channel of the output.

By setting different values for phase and gain for each source/destination pair the audio may be manipulated. For example, different gain values may be used to mono or leg-swap the incoming audio.

The following shows the fader settings (via `aCrosspointPathGain`) for different effects; *stereo* passes the audio unaltered, *swap* swaps the incoming left and right channels and *mono* places (M6) mono audio on both channels of the output.

Src	Dst	Gain		
		stereo	swap	mono
left	left	fullScale	mInfinity	-600
right	left	mInfinity	fullScale	-600
left	right	mInfinity	fullScale	-600
right	right	fullScale	mInfinity	-600

The number of input and output channels does not have to be the same, so the following models a block that takes a two channel stereo input to give a single channel mono output:



5.4.3 Audio clip player blocks

An audio clip player block shall have the following structure:

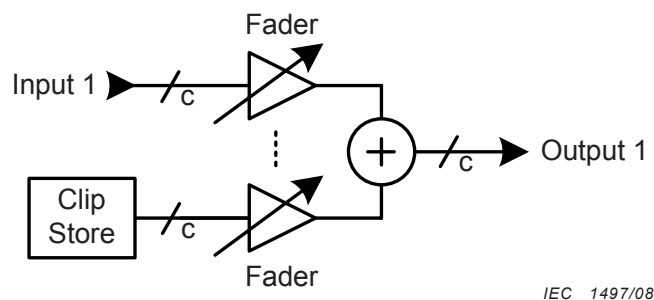


Figure 4 – Audio clip player block

where c is the number of channels on a connection. This represents a block that can either switch or mix a selected pre-stored audio sequence into an audio stream.

The group of objects in Table 7 shall be implemented by all compliant audio equipment that has a management model that incorporates one or more audio clip player blocks. The root node for these objects shall be

```
{ iso(1) standard(0) iec62379 audio(2) audioMIB(1) audioClipPlayer(4) }
```

This node shall be used as the block type identifier for audio clip player blocks.