

STANDARDS AND INFORMATION DOCUMENTS

AES64-2012
WITHDRAWN
(w2018)



**AES standard for
audio applications of networks -
Command, control, and
connection management for
integrated media
(Withdrawn 2018)**

Users of this standard are encouraged to determine if they are using the latest printing incorporating all current amendments and editorial corrections. Information on the latest status, edition, and printing of a standard can be found at:
<http://www.aes.org/standards>

AUDIO ENGINEERING SOCIETY, INC.
551 Fifth Ave., Suite 1225, New York, New York 10176, US.



The AES Standards Committee is the organization responsible for the standards program of the Audio Engineering Society. It publishes technical standards, information documents and technical reports. Working groups and task groups with a fully international membership are engaged in writing standards covering fields that include topics of specific relevance to professional audio. Membership of any AES standards working group is open to all individuals who are materially and directly affected by the documents that may be issued under the scope of that working group.

Complete information, including working group scopes and project status is available at <http://www.aes.org/standards>. Enquiries may be addressed to standards@aes.org

The AES Standards Committee is supported in part by those listed below who, as Standards Sustainers, make significant financial contribution to its operation.



Focusrite®



audio-technica



CLAIR



WEISS



METRIC HALO



This list is current as of
2018/9/01

AES standard for audio applications of networks - Command, control, and connection management for integrated media

Published by
Audio Engineering Society, Inc.
Copyright © 2012, 2018 by the Audio Engineering Society

Abstract

This standard for networked command, control, and connection management for integrated media is an IP-based peer-to-peer network protocol, in which any device on the network may initiate or accept control, monitoring, and connection management commands. The AES64 protocol has been developed around three important concepts: structuring, joining, and indexing. Every parameter is part of a structure, and control is possible at any of the levels of the structure, allowing for control over sets of parameters. Parameters can be joined into groups, thereby enabling control over many disparate parameters from a single control source. Every parameter has an index associated with it and, once discovered, this index provides a low bandwidth alternative to parameter control.

This document was WITHDRAWN on 2018-05-25.

An AES standard implies a consensus of those directly and materially affected by its scope and provisions and is intended as a guide to aid the manufacturer, the consumer, and the general public. The existence of an AES standard does not in any respect preclude anyone, whether or not he or she has approved the document, from manufacturing, marketing, purchasing, or using products, processes, or procedures not in agreement with the standard. Prior to approval, all parties were provided opportunities to comment or object to any provision. Attention is drawn to the possibility that some of the elements of this AES standard or information document may be the subject of patent rights. AES shall not be held responsible for identifying any or all such patents. Approval does not assume any liability to any patent owner, nor does it assume any obligation whatever to parties adopting the standards document. This document is subject to periodic review and users are cautioned to obtain the latest edition. Recipients of this document are invited to submit, with their comments, notification of any relevant patent rights of which they are aware and to provide supporting documentation.

Contents

Contents	2
Introduction	5
0 Preamble.....	5
0.1 Patents	5
0.2 Documentation conventions.....	6
1 Scope	6
2 Normative references	6
3 Definitions and abbreviations	7
4 Protocol overview	8
5 Message structure	9
5.1 General	9
5.2 Mapping tables.....	11
6 The AES64 message address block	12
6.1 Overview	12
6.2 The 7 address levels.....	12
6.3 Examples	15
6.4 The 'Wildcard' mechanism.....	16
6.5 Dummy values	17
7 Types of messages.....	17
7.1 Overview	17
7.2 Message type.....	17
7.3 Command Executive and Command Qualifier.....	20
8 Parameter Flags.....	25
8.1 Description of parameter flags.....	25
8.2 Parameter flags and user levels	27
8.3 The 'Select' capability	27
8.4 The 'Alert' capability	28
8.5 The alternate value capability	28
9 The Concept of 'Pushing'	29
9.1 Overview	29
9.2 Push list.....	29
10 Grouping Parameters	30
10.1 Overview	30
10.2 Mixed-relationship rules	31
10.3 Grouping mechanisms	31
11 Global units	35
12 Modifiers	35
12.1 Introduction	35
12.2 The Modifier concept.....	36
12.3 Value modifier	36
12.4 Level modifier.....	38
12.5 Event modifier	40
13 Desk Items	48

13.1 Overview	48
13.2 Desk item extraction.....	48
13.3 Desk item Information	49
13.4 Desk item layout.....	50
Annex A (normative): Level identifiers.....	51
A.1 Section block identifiers	51
A.2 Section type identifiers.....	53
A.3 Section number identifiers	54
A.4 Parameter block identifiers	54
A.5 Parameter block index	58
A.6 Parameter type	58
A.7 Parameter index	75
A.8 Value Format (VALFT).....	75
Annex B (normative): Desk item elements.....	78
Annex C (informative): Informative references	81
Annex D (normative): Registered values	82
D.1 IANA UDP port number	82
D.2 Manufacturer ID registration.....	82
D.3 Registered parameters.....	82
Annex E (informative): AES64 and XFN	83
Annex F (informative) Global units.....	84
F.1 Introduction	84
F.2 Range	84
F.3 Using Global Units with small controllers.....	85
F.4 Converting from AES64 parameters.....	86
Annex G (Normative) Error codes.....	88

Foreword

This foreword is not part of the AES64-2012 *AES standard for audio applications of networks - Command, control, and connection management for integrated media*.

This document grew out of an earlier XFN protocol and was developed within AES standards working group SC-02-12 under project AES-X170. The members of the writing group that developed this document in draft included R. Gurdan - leader of task group SC-02-12G - and R. Foss.

Richard Foss
Chair, working group SC-02-12 (June 2012)

WITHDRAWN

Note on normative language

In AES standards documents, sentences containing the word “shall” are requirements for compliance with the document. Sentences containing the verb “should” are strong suggestions (recommendations). Sentences giving permission use the verb “may”. Sentences expressing a possibility use the verb “can”.

AES standard for audio applications of networks - Command, control, and connection management for integrated media

Introduction

The AES working and task groups for audio applications of networks have up to now focused on connection management issues related to the routing of audio and MIDI data, as well as associated synchronization issues. This standard specifies an approach to the control and monitoring of professional audio devices within any IP-based audio and other media networks, and furthermore specifies how such an approach can be integrated with connection management.

0 Preamble

0.1 Patents

The Audio Engineering Society draws attention to the fact that it is claimed that compliance with this AES standard or information document may involve the use of the following patents:

European Patent PCT/EP/2008/062699 “Digital Multimedia Network with Hierarchical Parameter Control Protocol”, publication date 9 April 2009.

European Patent PCT/EP/2008/063195 “Digital Multimedia Network with Join Mechanism”, publication date 9 April 2009.

European Patent PCT/EP/2008/063255 “Digital Multimedia Network with Latency Control”, publication date 9 April 2009.

European Patent PCT/EP/2008/063256 “Data Stream Router”, publication date 9 April 2009.

European Patent PCT/EP/2008/063197 “An Apparatus and a Method for Modifying a Command Message in a Digital Multimedia Network”, publication date 9 April 2009.

The AES holds no position concerning the evidence, validity and scope of this patent right.

The holder of this patent right has assured the AES that it is willing to negotiate licenses under reasonable and non-discriminatory terms and conditions with applicants throughout the world. In this respect, the statement of the holder of this patent right is archived with the AES.

Information may be obtained from:

Mr Robby Gurdan: Universal Media Access Networks GmbH, Martinstraße 48-50, 40223 Düsseldorf, Germany

Prof Richard Foss: Computer Science Department, Rhodes University, Grahamstown 6140, South Africa

Attention is drawn to the possibility that some of the elements of this AES standard or information document may be the subject of patent rights other than those identified above. AES shall not be held responsible for identifying any or all such patent rights.

0.2 Documentation conventions

A Courier typeface may be used to identify computer listing examples to distinguish them from regular text.

Computer language examples are in the C language.

Following ISO convention, decimal points are conventionally shown as commas (,) unless an alternative, such as a period (.), is expressly stated here, with justification.

Values using hexadecimal notation are identified by a suffixed subscript 16; for example **8A₁₆**.

Case sensitivity: All parameters and identifiers listed in the specification are case sensitive.

1 Scope

This standard specifies:

1. A set of structured Internet Protocol-based messages and a specification of the messages that can be used to control single or groups of parameters, and to monitor these parameters. The parameters are associated either with typical signal processing functionality within audio and other media devices, synchronization between such devices, or connections between such devices.
 2. A definition of indexed parameters and a specification of the message format for such messages.
 3. A definition of peer-to-peer grouping and master-to-slave grouping of device parameters, and a specification of the messages required to implement such grouping.
 4. A definition of modifiers, and a specification of messages required to create and control such modifiers.
 5. A definition of desk items as a means to extract graphical user information from a device and allow for the control of the device via the extracted graphical controls.
- A definition of security levels, and a specification of messages required to set and get such levels.

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.

RFC 768 *User Datagram Protocol*, Internet Engineering Task Force, August 1980; www.ietf.org.

RFC 791 - *Internet Protocol*, Internet Engineering Task Force, September 1981; www.ietf.org.

RFC 1122 - *Internet Engineering Task Force, Requirements for Internet Hosts*, October 1989; www.ietf.org.

3 Definitions and abbreviations

3.1

User Datagram Protocol

UDP

a procedure for application programs to send messages to other programs with a minimum of protocol mechanism. See RFC 768.

3.2

Internet Protocol

IP

provides the functions necessary to deliver a package of bits (an internet datagram) from a source to a destination over an interconnected system of networks. See RFC 791

3.3

Parameter Identifier

an identifier (ID) that is unique for a particular parameter on the device that hosts it

3.4

Application

device-specific code that resides above the AES64 stack code and that utilizes the AES64 stack functionality in order to send and receive AES64 messages

3.5

Controller

a functional unit that resides on a typical device or workstation and that enables user control over devices on a network

3.6

Address Block

a structure that comprises 7 hierarchically arranged identifiers. The complete structure allows for the addressing of a particular parameter on a device

3.7

Level Identifier

an Identifier (ID) that is unique to a particular level of the address hierarchy

3.8

Global units

a generic unit that is used by AES64-based systems to hold parameter values and to pass values between parameters. Most AES64 parameters hold their current values in terms of Global units

3.9

Peer-to-peer group

a group of parameters that are joined in such a way that when the value of any member parameter changes, the values of all the other members of the group (peers) shall also change

3.10

Master slave group

a group of parameters that are joined in such a way that there is a single master and one or more slaves in the group. When the value of the master parameter changes, all the slave parameter values will change, but when the value of a slave parameter changes, it will have no effect on the other members of the group

3.11

Delta

a numerical difference between two absolute values of a parameter.

4 Protocol overview

AES64 specifies an IP-based peer-to-peer network protocol in which any device on the network can send or receive messages for connection management, control, and monitoring; and return similar messages in response. The size and capability of devices on the network will vary. Some devices will be large, and will incorporate extensive functionality, while other devices will be smaller and simpler.

Essential to this protocol is the fact that each parameter in a device can be addressed via a multi-level hierarchical structure that reflects the natural layout of the device. For example, there might be an input section on a mixing console that has a number of channel strips. Each channel strip might have a gain control, a block of equalizers, and a fader control. Each equalizer block will have its own structuring. Each of these groupings is considered to exist at a separate level. At the lowest level of any device structure are parameters.

An AES64 message shall access a parameter in a device by providing a hierarchical address that models the position of the parameter within the device, and thereby allows access to it. Each device shall implement an AES64 stack that will parse such a message structure, and thus be able to locate the parameter.

A “wild card” mechanism allows for control over multiple parameters. If the address of a message contains a wild card parameter at any particular level, then the addressing applies to all possibilities at that level. In this way a single command may affect a large number of parameters.

Apart from addressing a parameter via its hierarchical position in the device, it shall also be possible to address it via a unique identifier (ID). Each parameter has an ID that is unique for that parameter on the device that hosts it. This ID can be obtained by addressing the parameter and requesting it. Further messages can replace the hierarchical address, and simply provide the ID to access the parameter, thereby reducing message bandwidth.

The AES64 protocol allows parameters to be joined into groups. Each parameter can hold a list of other parameters on the same or different devices, to which it is joined. If the parameter is modified by a message, then that same message shall be directed at all parameters within its list. For example, a fader on a mixing console would typically have a parameter that represents its position. This parameter may be joined to fader parameters on other mixing consoles, and possibly to gain parameters on breakout boxes. When the single fader is moved, its group list would be scanned and messages sent to all joined parameters.

Lastly, the AES64 protocol incorporates the definition of “modifiers”, whereby any message can have its hierarchical message structure modified at any one of the levels. The modifier may for example increment the channel number level of all messages entering a device, and thereby effect immediate control over a second block of channels.

The AES64 protocol derives from an industry protocol named ‘XFN’. The code examples and some of the figures incorporate labels with the name XFN.

5 Message structure

5.1 General

5.1.1 Message

The primary purpose of AES64 messages is to get and set the various parameters of devices, and in so doing control and monitor the various capabilities of these devices.

Every AES64 message shall be contained within the payload area of a UDP datagram. The UDP header and payload area shall be contained within the payload area of an IP datagram that contains within its header a source and destination IP address.

The overall layout of an AES64 message is shown in Figure 1 below:

IP Header including source and destination IP address (unicast, broadcast, or multicast)
UDP Header including source and destination port (IANA reserved port no 7107)
AES64 message

Figure 1 - Overall Layout of AES64 message

The structure of an AES64 message has a number of components. The full layout of the message is given in figure 2 below, from most-significant byte (MSB) to least-significant byte (LSB). When setting the value of a parameter, the address block - shown here shaded - shall be followed by a value format field and a value field. A description of these various fields is given below.

MSB

Destination device ID (Source device node ID (total 32 bits) 128 bits)			
Destination device node ID (32 bits)			
Source device ID (128 bits)			
Source device node ID (32 bits)			
Source parameter ID (32 bits)			
User level (8 bits)	Message type (8 bits)	Sequence ID (32 bits)	
<i>sequence ID, continued</i>		Command Executive (8 bits)	Command Qualifier (8 bits)
X170 message address block (104 bits)			
Value Format (VALFMT)		Value (variable length)	

LSB

Figure 2 – AES64 message structure

5.1.2 Destination and source device ID

In order to identify source and destination devices at the AES64 level, a unique 128-bit device ID shall be assigned to each device. The device ID shall comprise:

- An 8-byte Extended Unique Identifier (EUI-64) as registered with IEEE-RA.
- A 3-byte Organizationally Unique Identifier (OUI) for the manufacturer of the device as registered with IEEE-RA.
NOTE The manufacturer of the device is not necessarily the manufacturer of the network interface.
- A 2-byte device ID that is unique amongst a given manufacturer's product range.
- A 3-byte reserved field.

A layout of the device ID is shown in figure 3:

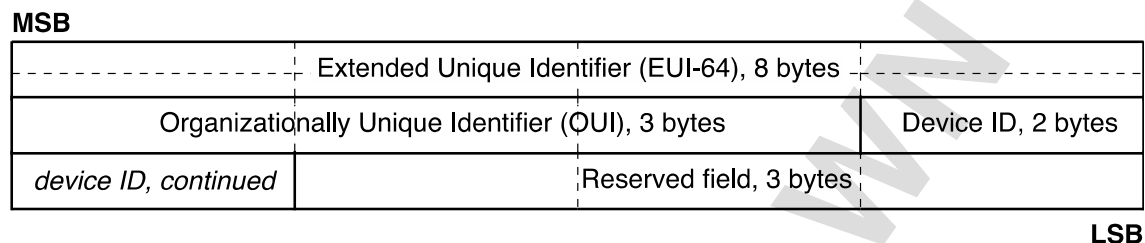


Figure 3 – AES64 device ID

Source and destination device ID's are accessible when the IP payload is being processed, and these device IDs shall be resolved to IP addresses.

5.1.3 Destination device node ID and source device node ID

Each device within an AES64 network will be uniquely addressed via its IP address.

A device may incorporate a number of units that function as complete devices, or a PC may act as an AES64 proxy for a number of non-AES64 devices, such as AV/C devices. In such cases, destination and source device node IDs are used to distinguish the individual units at the same IP address. A node ID shall comprise a 32-bit value that uniquely identifies a unit within a specific device identified by an IP address.

5.1.4 Source parameter ID

Every parameter within a device shall be uniquely identified by a parameter ID comprising a 32-bit value.

For example, unique parameter IDs will be used to identify a gain parameter on a mixing console, or a graphic slider on a workstation control window. These unique identifiers allow destination parameters to interact with source parameters unambiguously.

5.1.5 User level

The User Level field informs a parameter of the user's authority to make changes to that parameter. This is specified in detail in 8.2.

5.1.6 Message type

Certain AES64 messages will be requests, and others will be responses. Some AES64 messages will contain detailed hierarchical information to describe a parameter, while others will contain parameter IDs. The message type field indicates the nature of the message. See 7.2

5.1.7 Sequence ID

If an AES64 message source transmits a large number of messages, it needs to be able to relate each response message with its request message. The sequence ID provides this identification.

A sequence ID is a 32-bit value. The transmitter shall increment the sequence ID whenever a message is to be transmitted and will incorporate this sequence ID into the message. The receiver, when it responds to the message, will incorporate the sequence ID of the received message into its response.

5.1.8 Command Executive and Command Qualifier

Every AES64 message shall contain a command together with an indication (via the address block) of the parameter or group of parameters that the command relates to. Each parameter has a number of attributes such as its value, its state (identified by flags), its name, etc. The command structure is specified in 7.3

5.1.9 Address block

The address block provides the address of a parameter in a 7-level hierarchical format defined in 6.

5.1.10 Value format (VALFT)

This field serves to specify the format of the subsequent value field. Annex A.8 tables the various options for the **VALFT** field.

5.2 Mapping tables

There are many scenarios where parameter addresses need to be stored, and where these parameter addresses have to incorporate unique device identification. Furthermore, there will be circumstances where a failed device within a network needs to be replaced by a compatible device (unit replacement), or where an existing device needs to be assigned a new role (role replacement). In such cases, it is important that stored addresses relating to the original device now identify the new compatible device. A mapping table provides this functionality.

Each device shall implement locally a mapping table in the form of an indexed list. Each mapping-table index shall identify a device IP address and device ID. For each new device ID that it encounters, a device shall assign the next index to the device, and enter the device's IP address and device ID into the mapping table at that index position.

Each device shall build up its mapping table as device ID's and IP addresses become known as a result of messages sent to the device or group list updates sent to the device, for example (refer to 10).

When parameter addresses are stored within a device, the mapping table index for the device shall be stored rather than the device IP address or device ID. When a stored parameter address is used to address a particular parameter on the network, the index shall be used to access the appropriate device ID and IP address within the local mapping table of the device.

When a device is replaced on the network with a compatible device, the original device's device ID and IP address shall be replaced within the mapping tables of all devices on the network by the compatible device's device ID and IP address. In this way all stored parameter addresses relevant to the replaced device will correctly address the same parameters within the compatible device. A similar replacement shall take place if a current device assumes the role of another device on the network. All devices shall replace their mapping table entries for the displaced device with entries for the device with the new role.

6 The AES64 message address block

6.1 Overview

The AES64 message address block is used to address a parameter via a 7-level hierarchy. The structure of the address block resulted from the observation that any device can be viewed as a series of functional groupings, and that device parameters are positioned at the lowest level of such a series of groupings.

6.2 The 7 address levels

6.2.1 General

In the AES64 address block, these functional group levels are named as follows:

1. Section Block
2. Section Type
3. Section Number
4. Parameter Block
5. Parameter Block Index
6. Parameter Type
7. Parameter Index

In order to address a parameter, every level within the address block shall contain an identifier that indicates a particular predefined entity appropriate to that level. For each parameter in a device, a manufacturer shall choose such an entity for each level of the hierarchy. These functional groupings are specified below, and a set of identifiers appropriate for each level is provided in annex A.

The address-block structure is shown in more detail in Figure 4 below, shaded.

MSB

Destination device ID (Source device node ID (total 32 bits)128 bits)			
Destination device node ID (32 bits)			
Source device ID (128 bits)			
Source device node ID (32 bits)			
Source parameter ID (32 bits)			
User level (8 bits)	Message type (8 bits)	Sequence ID (32 bits)	
<i>sequence ID, continued</i>		Command Executive (8 bits)	Command Qualifier (8 bits)
Section block (8 bits)	Section type (8 bits)	Section number (24 bits)	
<i>section number, continued</i>	Parameter block (8 bits)	Parameter block index (24 bits)	
<i>parameter block index, continued</i>	Parameter type (16 bits)		Parameter index (16 bits)
<i>parameter index, continued</i>	Value Format (VALFMT)	Value (variable length)	

LSB

Figure 4 – AES64 message with address block detail

6.2.2 Section Block

This is the highest functional group. Any device can be considered to comprise a number of sections. This top-level grouping shall identify the high-level section block within which the parameter resides.

An example, could include an Input section, output section, input matrix section, output matrix section, etc. See also annex A.1.

6.2.3 Section Type

This can be viewed as a sub-group within the section block, and shall be used to differentiate components of the section block.

For example, within the input section block of a mixing console, there might be various types of inputs - microphone inputs, line inputs, tape inputs, etc. See also annex A.2.

6.2.4 Section Number

The section number shall be used to indicate an interface number or a channel number where appropriate.

Most professional audio parameters will be related to an audio channel. These parameters will be responsible for processing a channel or routing a channel. It is important to be able to trace a channel throughout a multi-

device network, and this capability is facilitated by ensuring that each parameter structure includes the channel number at level 3. See also annex A.3.

6.2.5 Parameter Block

Parameters are often clustered into parameter groups, and these parameters as a group allow for the processing and routing of audio channels. The parameter block shall identify such parameter groups. See also annex A.4.

An example of a parameter block could include a block of equalizers that allow for wide ranging equalization of an audio channel.

6.2.6 Parameter Block Index

This grouping allows for differentiation of similar components within a parameter block.

An example could include equalization sub-groupings, where each of the sub-groupings has a similar function. In the case of a parametric equalizer parameter block, there might be parametric equalizer 1 with 4 full parametric bands, and parametric equalizer 2 with 2 full parametric bands. The parameter block index would serve to distinguish between these two cases.

The parameter block index is also used to enable the use of manufacturer-specific parameters. This usage is applicable when the Parameter Block is set to “F2” (Manufacturer Parameter Block). The Manufacturer Parameter Block can set parameter block indices that are specific to each manufacturer. See also annex A.5, and annex D.

6.2.7 Parameter Type

This level shall indicate the type of the parameter being accessed, examples being gain, low frequency, threshold, etc. See annex A.6.

6.2.8 Parameter Type index

This level provides for accurate addressing in cases where there is more than 1 parameter of the same type that processes the audio of a channel. The occurrence of multiple gain parameters within a channel is an example. See annex A.7.

6.3 Examples

Figure 5 below shows an AES64 message example where the levels of the address block have been assigned appropriate identifiers. The parameter in this case is the gain for a channel on a mixing console.

MSB

Destination device ID (Source device node ID (total 32 bits)128 bits)			
Destination device node ID (32 bits)			
Source device ID (128 bits)			
Source device node ID (32 bits)			
Source parameter ID (32 bits)			
User level (8 bits)	Message type (8 bits)	Sequence ID (32 bits)	
<i>sequence ID, continued</i>		Command Executive (8 bits)	Command Qualifier (8 bits)
ID=01₁₆ Section block = INPUT_SIGNAL	ID=D1₁₆ Section type = AUDIO	ID=000001₁₆ Channel number = "1"	
<i>section number, continued</i>	ID=11₁₆ Param. block = DIGITAL_GAIN	ID=000001₁₆ Index-entry= "1"	
<i>parameter block index, continued</i>	ID=0201₁₆ Parameter type = GAIN		ID=0000₁₆ Index entry = "1"
<i>parameter index, continued</i>	Value Format (VALFMT)	Value (variable length)	

LSB

Figure 5 - Address block entry for a gain parameter

In tabular form, the series of values that make up the 7-level address of the gain parameter in this example, are:

Address layer	Value	Label
1. Section Block	01 ₁₆	INPUT_SIGNAL
2. Section Type	D1 ₁₆	AUDIO
3. Section Number	000001 ₁₆	CHANNEL_NUMBER = "1"
4. Parameter Block	11 ₁₆	DIGITAL_GAIN
5. Parameter Block Index	000001 ₁₆	INDEX_ENTRY = "1"
6. Parameter Type	0201 ₁₆	GAIN
7. Parameter Index	0000 ₁₆	INDEX_ENTRY = "1"

Whether the addressed parameter is retrieved, set to a value, or some other action is performed, will depend on the nature of the command as specified in the Command Executive and Command Qualifier fields (see 5.1.8, 7.3).

6.4 The 'Wildcard' mechanism

In place of a single parameter at any level, a special value may denote a wildcard that identifies a set of parameters.

Thus in the example in section 6.3, the gains on all channels of a device could be modified by a single AES64 message if the message contains the following address with a section-number wildcard:

Address-block element	Value	Label
1. Section Block	01 ₁₆	INPUT_SIGNAL
2. Section Type	D1 ₁₆	AUDIO
3. Section Number	3FFFFFF ₁₆	ALL_CHANNEL_NUMBERS
4. Parameter Block	11 ₁₆	DIGITAL_GAIN
5. Parameter Block Index	000001 ₁₆	INDEX_ENTRY = "1"
6. Parameter Type	0201 ₁₆	GAIN
7. Parameter Index	0000 ₁₆	INDEX_ENTRY = "1"

The wildcard in the above example indicates that for the given section block and section type, the parameter with the given parameter block and parameter block index and parameter type and parameter index on every channel shall be modified.

In the case of some levels, there are sub-categories within the levels, and specific wild card identifiers have been allocated to select all members of these sub-categories. For example, in the case of the 'Section Number' level, there are the 'Channel Number' and 'Stream Number' sub-categories. The wild card identifiers 3FFFFFF and 80FFF can be used to select all members of the respective sub-categories. See annex A.3.

It is possible for the wildcard mechanism to be used in conjunction with a broadcast IP address or multicast IP address, thereby allowing for control over multiple devices. Analogous to the IP mechanism, a certain device ID will be a broadcast address, and there is a certain range of device ID's allocated for multicast addressing. A particular device shall be addressed via a single unique device ID, a broadcast device ID, and potentially more than one multicast device ID.

In the above example, all gains on all channels of all devices in a network could be controlled via a single AES64 message. A wildcard may appear at more than one level within an address block. Thus in the above example, if a wildcard were to be introduced at the section type level, then access to gain parameters would be widened to include all section types within the input section block.

The wildcard mechanism can be used to have a wide-ranging effect on a large number of device parameters within a network. However it does depend on three things:

- Parameters that conform to a fixed level-based addressing scheme
- Consistent grouping at each level
- The use of pre-defined identifiers for the groupings at each level

Manufacturers should/shall assign addresses to parameters in a consistent fashion.

As shown in annex A, there are already a substantial number of published identifiers at the various levels. If the current identifiers do not fulfill the addressing requirements of device parameters, then manufacturers can request additional identifiers as described in annex D.

6.5 Dummy values

A parameter address can be specified without utilizing all 7 levels. This will particularly apply to simple devices that do not have complex hierarchical functional groupings. In this case, 'dummy' values shall be placed in the unnecessary levels.

For example, if there is only a single parameter of the 'gain' type on a particular input channel of a mixing console, then the parameter index shall contain the value '0000'.

If a device receives a message with a parameter address that does not exist on the device, then the device shall discard the message. If a device receives a message with a valid parameter address, but invalid value in the Value field, then the device shall transmit a response, where the Value Format field contains the value 27_{16} , indicating an error (see annex A.8.3), and the value field contains the value 4, indicating 'Malformed data' (see annex G).

7 Types of messages

7.1 Overview

Every AES64 message will either be a request sent from one device to another, or will be a response from a device after a request has been made. Some message requests require a response, while others do not.

There are three fields within an AES64 message that together indicate the type of message that is being transmitted. These are the Message Type, Command Executive, and Command Qualifier fields.

7.2 Message type

7.2.1 General

Table 1 describes the various AES64 message types. The form of the message using a full address block is shown in 3.

Table 1 - Message type definitions

Hexadecimal value	Message type definitions
00_{16}	Full address block requiring a response
01_{16}	Full address block requiring no response
02_{16}	Indexed message requiring a response
03_{16}	Indexed message requiring no response
04_{16}	Response message
05_{16}	Full address block with second destination and requiring a response
06_{16}	Full address block with second destination and requiring no response
07_{16} to FF_{16}	Reserved for future standardization

7.2.2 Indexed message

The parameter being addressed may also be addressed with a 32-bit unique parameter identifier using an indexed message. A parameter identifier is an integer value that uniquely identifies a parameter within a device.

Figure 6 below shows the format of an indexed message. The address block has been replaced by a destination parameter ID. All other fields of the AES64 message remain the same.

MSB

Destination device ID (Source device node ID (total 32 bits)128 bits)			
Destination device node ID (32 bits)			
Source device ID (128 bits)			
Source device node ID (32 bits)			
Source parameter ID (32 bits)			
User level (8 bits)	Message type (8 bits)	Sequence ID (32 bits)	
<i>sequence ID, continued</i>		Command Executive (8 bits)	Command Qualifier (8 bits)
Destination parameter ID (32 bits)			
Value Format (VALFMT)	Value (variable length)		

LSB

Figure 6 - Indexed message format

7.2.3 Response message

For many AES64 messages, a response is expected from the recipient of the request (refer to Table 1). The response message is the same, regardless of whether the request contained a full address block or parameter ID. The format of the response message shall be as shown in figure 7 below. Note that a response message will not contain the Command Executive and Command Qualifier fields. The sequence ID allows the device that transmitted the request to determine which request the response belongs to (see also 5.1.7).

MSB

Destination device ID (Source device node ID (total 32 bits)128 bits)			
Destination device node ID (32 bits)			
Source device ID (128 bits)			
Source device node ID (32 bits)			
Source parameter ID (32 bits)			
User level (8 bits)	Message type (8 bits)	Sequence ID (32 bits)	
<i>sequence ID, continued</i>		Value Format (VALFMT)	Value (variable length)

LSB

Figure 7 - Response message

7.2.4 Second-destination messages

There are cases where it is necessary to specify a second destination address, for example when a message with a particular value is sent to a parameter, and it is required that this message undergoes modification and retransmission to a further parameter. Clause 12 describes the ‘modifier’ concept, and subclause 12.4 describes the ‘level modifier’, which requires this second destination capability.

Figure 8 shows the message layout for a message that incorporate a second (or “2nd”) destination. There may only be a single level modifier between a source and destination parameter.

MSB

Destination device ID (128 bits)			
Destination device node ID (32 bits)			
Source device ID (128 bits)			
Source device node ID (32 bits)			
Source parameter ID (32 bits)			
2nd Destination device ID (128 bits)			
2nd Destination device node ID (32 bits)			
User level (8 bits)	Message type (8 bits)	Sequence ID (32 bits)	
<i>sequence ID, continued</i>		Command Executive (8 bits)	Command Qualifier (8 bits)
Section block (8 bits)	Section type (8 bits)	Section number (24 bits)	
<i>section number, continued</i>	Parameter block (8 bits)	Parameter block index (24 bits)	
<i>parameter block index, continued</i>	Parameter type (16 bits)		Parameter index (16 bits)
<i>parameter index, continued</i>	2nd Section block (8 bits)	2nd Section type (8 bits)	Section number (24 bits)
<i>2nd section number, continued</i>		2nd Parameter block (8 bits)	2nd Parameter block index (24 bits)
<i>2nd parameter block index, continued</i>		2nd Parameter type (16 bits)	
2nd Parameter index (16 bits)		Value Format (VALFMT)	Value (variable length)

LSB

Figure 8 - Full address block message with second destination

7.3 Command Executive and Command Qualifier

A command shall comprise two components. A command executive component shall indicate the fundamental nature of the command (get, set, etc); while the command qualifier component shall direct the command executive to a particular attribute of a parameter, or a particular grouping of parameters (such as a selection group).

In this document a specific command will be shown in single quotes as ‘Executive Qualifier’, for example: ‘GET VAL’.

Table 2 specifies the ‘command executives’ together with a brief explanation of their respective functions.

Table 2 - Command Executives

Hex ID	Command	Short explanation
00	GET	Get one or more data values
01	SET	Set one or more data values
02	ACT	Perform an action
03	JOIN	Join a parameter to a group
04	UNJOIN	Detach a parameter from a group
05	CREATE	Create a structure such as a list
06	DELETE	Delete a structure such as a list
07	SAVE	Save a structure such as a list
08 to FF	<i>reserved</i>	<i>reserved for future standardization</i>

Table 3 specifies the range of command qualifiers that can be coupled with the command executives.

Table 3 - Command Qualifiers

Hex ID	Command Qualifier	Description
00	VAL	Refers to the parameter’s value. This will cause the GET and SET command executives to get or set the value of the addressed parameter. The VALFT field of the address block will determine the size and type of the value.
01 to 06	<i>reserved</i>	<i>reserved for future standardization</i>
07	PUSH	Register a push listener with a parameter. Every parameter shall have a ‘Push List’. The push list contains a list of full addresses of AES64 parameters. This PUSH qualifier is used to add a parameter to the push list of a second parameter. Push functionality is described in 9.
08	FLAG	Refers to the various flags of a parameter, and allows for the getting and setting of flag values. Each parameter has a number of flags associated with it. The flags indicate, for example, whether the parameter can be written to, whether it has security enabled, etc. This command qualifier allows a parameter’s flags attribute to be read from or written to. Further information regarding parameter flags is provided in 8.
09	<i>reserved</i>	<i>reserved for future standardization</i>
0A	NAME	Allows for the querying of a parameter’s name when used with a GET command executive.

0B	VTBL	Value table. Refers to the value names of a parameter. For display purposes, it should be possible to display every value that a parameter can take on as a meaningful character string. For example, if there is a graphical representation of a slider, and the slider controls a gain parameter, then there should be decibel indicators on this slider. This command qualifier allows one to obtain such a list of meaningful character strings from a parameter.
0C	ID	Allows for the querying of a parameter's ID when used with a GET command executive.
0D	SNP	Refers to a snapshot of a device's parameters. This qualifier relates to snapshot (state store/recall capabilities). The snapshot model is described in 7.4.
0E to 10	<i>reserved</i>	<i>reserved for future standardization</i>
11	PTPGRP	Refers to the peer-to-peer group of a parameter – see 10
12	SLVGRP	Refers to the slave group of a parameter – see 10
13	MSTGRP	Refers to the master group of a parameter – see 10
14	GROUP_VALUE	Group value. Refers to the value of a parameter within a group – see 10
15 to 29	<i>reserved</i>	<i>reserved for future standardization</i>
2A	CLA	Refers to child level aliases, which provide meaningful names to the various levels in the parameter hierarchy. Associated with every level node in the AES64 tree is an alias string. These aliases are set when the levels are created at initialization time. The CLA qualifier shall be used with a GET command executive to get all of the child level aliases for a specified parent level.
2B to 2C	<i>reserved</i>	<i>reserved for future standardization</i>
2D	MASTER	Used to add a group of master parameters to a slave parameter – refer to 10
2E	SLAVE	Used to add a group of slave parameters to a master parameter – refer to 10
2F	PEER	Used to add a group of peer parameters to another peer parameter – refer to 10
30	MASTER_OFF	Used to remove a group of master parameters from a slave parameter – refer to 10
31	SLAVE_OFF	Used to remove a group of slave parameters from a master parameter – refer to section 10
32	PEER_OFF	Used to remove a group of peer parameters from another peer parameter – refer to 10
33 to 34	<i>reserved</i>	<i>reserved for future standardization</i>
35	PTP	Used with the JOIN command to join two peer to peer parameters – refer to 10
36	MSTSLV	Used with the JOIN command to join a slave to a master parameter – refer to 10
37	PUSH_OFF	The PUSH_OFF qualifier shall be used to remove a parameter from the push list of a second parameter.
38	DATA_BLOCK	This qualifier shall be used to indicate to a receiving parameter that there is a block of data being sent to it. It is used in the context of 'pushing', when a parameter wants to push a block of data to a second parameter. An example of such a block of data is a block of meter values.

39 to 4F	<i>reserved</i>	<i>reserved for future standardization</i>
50	BLACKLIST	Used to indicate that a parameter on a peer-to-peer list is 'blacklisted'. Refer to 10.3.5, also to 8.1 for device blacklist.
51	MODIFIER	This serves to indicate that a command such as CREATE relates specifically to a modifier. Refer to section 12.
52	EVENT	This serves to indicate that a command relates specifically to a timed event. Refer to 12.
53	TARGETED_PUSH	Every parameter shall have a 'targeted push list'. Each entry of the targeted push list contains the full address of an immediate Destination (for example a modifier Input Value Parameter), and the full address of a second destination (for example the destination address that the Output Value Parameter of a modifier will push to). Refer to 12.4.
54	TARGETED_PUSH_OFF	The TARGETED_PUSH_OFF qualifier shall be used to remove a parameter from the targeted push list of a second parameter.
55	TARGETED_VAL	This qualifier indicates to a receiving parameter that the message incorporates a Destination address as well as a value. Refer to 12.4.
56	MAPTABLE	This qualifier relates to the mapping table within a device as described in 5.1 and 7.3.2.
57	USR_LVL_LISTNO	This qualifier relates to the user-level list that a parameter will access to determine the parameter flag settings appropriate to the user level provided in a message sent to the parameter. Refer to 8.2.
58	SEL	This qualifier relates to the 'Select' group on a device. Further information regarding the Select group can be found in 8.3.
59 to FF	<i>reserved</i>	<i>reserved for future standardization</i>

7.4 The Snapshot model

Each device shall determine a set of parameters that it wishes to save as a snapshot.

To save a snapshot, a controller shall send a '**SAVE SNP**' command to a device, where the value field of the message contains the ID of a device snapshot. The message shall be addressed to the snapshot parameter on the device, with the following address block:

Address-block element	Value	Label
1. Section Block	C1 ₁₆	USER
2. Section Type	00 ₁₆	CONTROLLER
3. Section Number	000000 ₁₆	<i>dummy value</i>
4. Parameter Block	F1 ₁₆	PRESET
5. Parameter Block Index	000000 ₁₆	<i>dummy value</i>
6. Parameter Type	0E64 ₁₆	SNAPSHOT
7. Parameter Index	0000 ₁₆	<i>dummy value</i>

To recall a snapshot, a controller shall issue a '**SET VAL**' command addressed to the same snapshot parameter, where once again the value field contains the ID of the snapshot within the device. This is shown diagrammatically in figure 9 below.

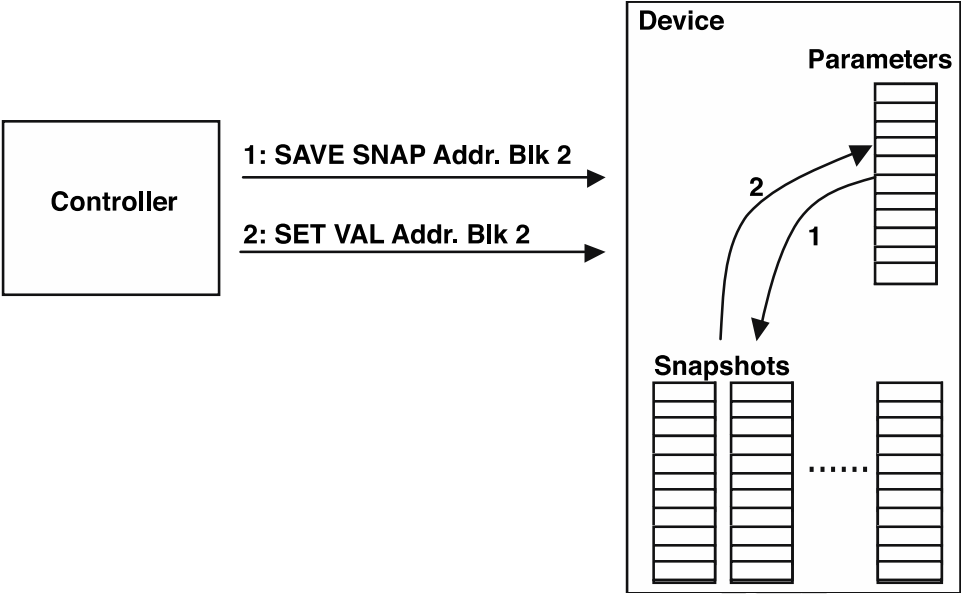


Figure 9 - Saving and recalling snapshots

7.5 The Mapping Table mechanism

As indicated in 5.2, each device contains a mapping table that maps a local index to an IP address and device ID. Table 4 below indicates the nature of the mapping table.

Table 4 – Mapping Table – index to Device ID and IP address

Index	Device ID	IP address
1	128-bit ID	146.234.52.1
2	128-bit ID	146.234.52.23
.	.	.
.	.	.
n	128-bit ID	IP address

Each entry in the mapping table shall comprise a local index, the device ID, and the device IP address. The local index shall be assigned by the device, and it shall be unique to the device. The mapping table is local to a device, and so the number of entries in the table is implementation-specific.

When any device starts up, it shall broadcast its device ID and IP address within a ‘SET MAPTABLE’ message with the following parameter address block:

Address-block element	Value	Label
1. Section Block	F1 ₁₆	DEVICE_INFO_CONFIG
2. Section Type	E0 ₁₆	CONFIGURATION
3. Section Number	000000 ₁₆	dummy value
4. Parameter Block	96 ₁₆	NETWORK_MAPPING_TABLE
5. Parameter Block Index	000000 ₁₆	dummy value
6. Parameter Type	0F30 ₁₆	CONFIG_TABLE_ADD
7. Parameter Index	0000 ₁₆	dummy value

The value field shall contain the device ID and device IP address of the new entry. Any other device receiving this broadcast message, shall add a unique local index to its mapping table, and associate the device ID and IP address with this local index.

When a controller starts up, it shall discover the IP addresses of all devices on a network by broadcasting a 'GET VAL' command with the following parameter address:

Address-block element	Value	Label
1. Section Block	F1 ₁₆	DEVICE_INFO_CONFIG
2. Section Type	E2 ₁₆	INTERFACE_CONFIG
3. Section Number	FFFFFF ₁₆	ALL_NUMBERS
4. Parameter Block	FF ₁₆	ALL
5. Parameter Block Index	FFFFFF ₁₆	ALL_INDICES
6. Parameter Type	0F00 ₁₆	IP_ADDRESS
7. Parameter Index	0000 ₁₆	<i>dummy value</i>

For each discovered device, the controller shall get the device ID of the device. The controller shall assign a local index and create a new entry in the device's mapping table, as shown in table 4 above.

A device that does not have controller capabilities shall also build up a mapping table.

A device-mapping table may also be built up via the join mechanism. In fulfilling a request to parameter A to join to parameter B, parameter A shall request all parameter B's group entries. Parameter B shall provide details of each entry in its group, where these details include the device ID of the parameter. If parameter A does not have an entry for a particular device in its mapping table, it shall assign a new local index, and broadcast a request for the IP address of the device with the given device ID. This message shall carry a 'GET MAPTABLE' command, with the following parameter address block, and with a value field that contains the device ID:

Address-block element	Value	Label
1. Section Block	F1 ₁₆	DEVICE_INFO_CONFIG
2. Section Type	E0 ₁₆	CONFIGURATION
3. Section Number	000000 ₁₆	<i>dummy value</i>
4. Parameter Block	96 ₁₆	NETWORK_MAPPING_TABLE
5. Parameter Block Index	000000 ₁₆	<i>dummy value</i>
6. Parameter Type	0F30 ₁₆	CONFIG_TABLE_ADD
7. Parameter Index	0000 ₁₆	<i>dummy value</i>

The response to this message shall contain the IP address of the device newly entered into the mapping table.

Each group entry shall not contain the device ID and device IP address of a grouped parameter's device, but rather the local index assigned to this device. Device ID and device IP information shall be accessible from the device's mapping table via the local index.

This mechanism reduces the memory used to store group and other lists, and also enables efficient updating. When a device's IP address is changed, or when a device is replaced, the original device ID and device IP address shall be replaced by the new device ID and device IP address within the mapping tables of each device on the network. The local indices in group lists and other tables in all devices throughout the network will then correctly access, via the updated mapping tables, the updated or replaced device.

7.6 Version information

The AES64 protocol will typically be implemented by an AES64 stack within workstations and media devices. It is important for a controlling device to know the version number of the stack of a device that it is controlling, since the messaging capability of a stack may change from one version to the next.

An AES64 device can obtain the version number of the stack of another device by accessing the version-number parameter of that device. This can be done by transmitting a 'GET VAL' command to the device with the following address block:

Address-block element	Value	Label
1. Section Block	F1 ₁₆	DEVICE_INFO_CONFIG
2. Section Type	E0 ₁₆	CONFIGURATION
3. Section Number	000000 ₁₆	dummy value
4. Parameter Block	97 ₁₆	NETWORK_STACK
5. Parameter Block Index	000000 ₁₆	dummy value
6. Parameter Type	0E70 ₁₆	VERSION
7. Parameter Index	0000 ₁₆	dummy value

The version number, a 32-bit value, shall be returned by the device. The most significant 16 bits shall contain the major version number and the least significant 16 bits shall contain the minor version number. The version number relates to a specific revision of this standard.

8 Parameter Flags

8.1 Description of parameter flags

Each parameter has a number of flags associated with it. These flags indicate various states that the parameter can occupy. A flag is assigned a single bit within a 32-bit flags register. This bit can be either 0 or 1, indicating that the flag is either not set or set, respectively. The flags register can be accessed and modified by using the **FLAG** command qualifier, as set out in 7.3.

Table 5 below lists all the available parameter flags, and a description of the role played by each of the flags:

Table 5 - Parameter flags

Bit Count, (decimal)	Function	Description
lsb 0	<i>SEL</i>	Any parameter in a device can be made part of a single "selection" group. The SEL command qualifier shall be used to access the parameters of the selection group. For example a 'SET SEL' command shall be used to set all the parameters of the selection group to a particular value. The 'GET SEL' command shall be used to get all the selection group parameters, that is, the address blocks of all the parameters of the selection group.
1	<i>ALERT</i>	The ALERT flag can be set when a parameter fulfills a particular critical condition, such as a temperature value on an amp that is inordinately high. As with all the flags, a parameter's ALERT flag can be polled via the 'GET FLAGS' command. A controller can also be alerted to an "alert" condition on a device via an alert parameter. This is described in more detail in section 8.4.
2	<i>READ</i>	The READ flag indicates whether a parameter may be read or not.
3	<i>WRITE</i>	The WRITE flag indicates whether a parameter may be written to or not.
4	<i>VALUE_PROTECT</i>	The VALUE_PROTECT flag indicates whether the value of a parameter may be overwritten via a snapshot.
5	<i>VALUE_TABLE_PROTECT</i>	The VALUE_TABLE_PROTECT flag indicates whether the value table of a parameter may be overwritten via a snapshot.
6	<i>HIDE</i>	If a parameter is hidden, then it will not be accessible for enumeration by