

STANDARDS AND INFORMATION DOCUMENTS

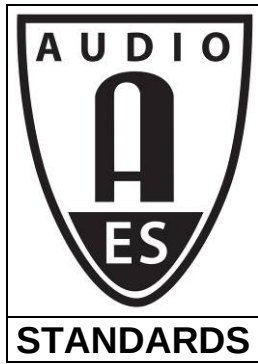
AES70-1-2018
(Rev. AES70-1-2015)



AES standard for audio applications of networks - Open Control Architecture - Part 1: Framework

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 Avenue, New York, NY 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®



This list is current as of 2019/1/12

AES standard for audio applications of networks - Open Control Architecture - Part 1: Framework

Published by
Audio Engineering Society, Inc.
Copyright © 2015, 2018 by the Audio Engineering Society

Abstract

AES70 defines a scalable control-protocol architecture for professional media networks. AES70 addresses device control and monitoring only; it does not define standards for streaming media transport. However, the Open Control Architecture (AES70) is intended to cooperate with various media transport architectures.

AES70 is divided into a number of separate parts. This Part 1 describes the models and mechanisms of the AES70 Open Control Architecture. These models and mechanisms together form the AES70 Framework. This document should be read together with Part 2, Class Structure and Part 3, TCP/IP communications protocol.

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. 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. This document is subject to periodic review and users are cautioned to obtain the latest edition.

Audio Engineering Society Inc., 551 Fifth Avenue, New York, NY 10176, US.

www.aes.org/standards standards@aes.org

2019-1-14 printing

 **Documents**
Document Sharing Hub

Contents

0.	Introduction	6
0.1.	General	6
0.2.	Architectural goals and constraints.....	6
0.3.	Document conventions	8
1.	Scope.....	8
2.	Normative References	8
3.	Terms, definitions and abbreviations	9
3.1.	AES70-compliant interface AES70 interface	9
3.2.	Media device.....	9
3.3.	Non-media device.....	9
3.4.	AES70 device Device	9
3.5.	Non-AES70 device	9
3.6.	Device model	9
3.7.	Controller	9
3.8.	Control	9
3.9.	Control protocol	9
3.10.	Media stream	9
3.11.	Open Control Protocol OCP	10
3.12.	OCP.1	10
3.13.	Protocol data unit PDU	10
3.14.	Object Number ONo	10
3.15.	Precision Time Protocol (PTP).....	10
3.16.	Program or AES70 Program	10
3.17.	Task AES70 Task	10
3.18.	Voltage-controlled amplifier VCA	10
3.19.	Binary Large Object BLOB.....	10
3.20.	Robustness	10
3.21.	Availability	10
3.22.	Alignment level.....	10
4.	Top level design	10
4.1.	General	10
4.2.	Object orientation.....	11
4.2.1.	General.....	11
4.2.2.	Classes.....	11
4.2.3.	Instantiation of classes	16
4.3.	Messages	16
4.3.1.	Basic mechanism	16
4.3.2.	Message delivery services	17
5.	Device model	17
5.1.	Device configurability.....	17
5.2.	Object addressing.....	18
5.3.	Device model elements	19
5.3.1.	General.....	19
5.3.2.	Managers, Workers, and Agents.....	19
5.4.	Worker classes	20
5.4.1.	Actuators	20
5.4.2.	Sensors	20
5.4.3.	Blocks	21
5.4.4.	Matrices	25

5.5.	Agent classes	30
5.5.1.	General	30
5.5.2.	OcaGrouper	30
5.5.3.	OcaRamper	35
5.5.4.	OcaTask	36
5.5.5.	OcaNumericObserver and OcaNumericObserverList	36
5.5.6.	OcaLibrary	36
5.5.7.	OcaMediaClock3	36
5.5.8.	OcaTimeSource	36
5.5.9.	OcaPhysicalPosition	37
5.5.10.	OcaEventHandler	37
5.6.	Network classes	37
5.7.	Manager classes	37
5.8.	Standard Object Numbers (ONo)	38
5.9.	Object text identification	38
5.10.	Constructing objects	38
5.10.1.	General	38
5.10.2.	Block Factories	39
5.11.	Deleting objects	39
6.	Events and subscriptions	39
6.1.	Subscriptions, events, emitters and notifications	39
6.1.1.	General	39
6.1.2.	Reliable or Fast subscriptions	39
6.1.3.	Lightweight subscriptions	40
6.1.4.	Subscription deletion	40
6.1.5.	Subscription aggregation	40
6.2.	Subscription event handler	41
6.3.	The PropertyChanged event	41
6.4.	Use of numeric observers	42
7.	Networking and connection management	42
7.1.	General	42
7.2.	Connection Management Versions	43
7.3.	The CM3 object model	43
7.3.1.	Classes	43
7.3.2.	Structure	45
7.4.	AES70 Adaptations (informative)	47
8.	Time	47
8.1.	Format	47
8.2.	PTP Epoch	47
8.3.	UTC time and the NTP time format	48
9.	Physical Position Coordinate Systems	48
9.1.	Robotic coordinates	49
9.2.	ITU coordinates	49
9.2.1.	ITU object-based polar coordinates	49
9.2.2.	ITU object-based Cartesian coordinates	49
9.2.3.	ITU scene-based polar coordinates	49
9.2.4.	ITU scene-based cartesian coordinates	50
9.3.	Navigation coordinates	50
10.	Libraries	50
10.1.	ParamSet and Patch libraries	51
10.1.1.	ParamSets	51

10.1.2.	Patches	52
10.2.	Program Libraries.....	52
10.3.	Proprietary libraries	52
11.	Tasks	52
12.	Sessions.....	53
13.	Security	53
14.	Concurrency control.....	53
15.	Reliability	54
15.1.	General	54
15.2.	Availability	55
15.2.1.	Keep-alive messages	55
15.2.2.	Efficient reinitialization	55
15.3.	Robustness	55
16.	Device reset	55
17.	Firmware and software upgrade.....	56
17.1.	Update Types.....	56
17.2.	Update Modes.....	57
17.3.	Mechanisms	57
17.3.1.	Active Updating.....	57
17.3.2.	Passive Updating.....	57
Annex A.	(Informative) – Actuator Example	59
A.1.	General	59
A.2.	Properties, Methods, and Events	59
Annex B.	(Informative) – Block examples	61
A.	Simple microphone channel	61
B.	Two-channel microphone mixer	61
C.	Mixer using nested blocks	62
Annex C.	(Informative) – Other Media Network Control Standards.....	63
A.	SMPTE ST 2071 - Media Device Control.....	63
B.	Architecture for Control Networks (ACN)	63
C.	Open Sound Control (OSC).....	63
D.	Ember+	63
Annex D.	Bibliography.....	64

Foreword

This foreword is not part of this document, AES70, *AES standard for audio applications of networks - Open Control Architecture - Part 1: Framework*.

This document is a member of the three-document set that defines AES70, the Open Control Architecture (OCA). AES70-11 defines the architectural framework for AES70. Other parts define the control repertoire and the specific protocols used.

The development project for this standard was originally proposed by the Open Control Architecture Alliance (OCA Alliance) and initiated in October 2012 as project X210 to be developed in task group SC-02-12-L. The OCA Alliance also contributed to the task-group working draft and, as a result, there are various references to "OCA" in the protocol, in order to maintain compatibility with implementations already in the field.

The members of the writing group that developed this document in draft are: J. Berryman, H. Hamamatsu, T. Head, S. Jones, M. Lave, N. O'Neill, M. Renz, M. Smaak, G. van Beuningen, S. van Tienen, E. Wetzell.

J. Berryman led the task group.

Richard Foss
Chair, working group SC-02-12
2015-11-12

Foreword to the 2018 edition

The members of the writing group that drafted this version of the document are: F. Bergholtz, J. Berryman, A. Gödeke, J. Grant, A. Kuzub, M. Lave, G. Linis, S. Price, M. Renz, A. Rosen, S. Scott, G. Shay, P. Stevens, P. Treleaven, S. van Tienen, and E. Wetzell. As well, the writing group was materially assisted by contributions from T. de Brouwer, B. Escalante, S. Jones, M. Smaak, and M. Versteeg.

J. Berryman led the writing group.

Morten Lave
Chair, working group SC-02-12
2018-12-20

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 - Open control architecture - Part 1: Framework

0. Introduction

0.1. General

This document describes the models and mechanisms of the AES70 Open Control Architecture (AES70) for the control and monitoring of media networks. These models and mechanisms together form the AES70 Framework.

AES70 is for system control and monitoring only, and may be integrated with any streaming program transport protocol scheme, as long as the underlying communication network is capable of carrying AES70 control and monitoring traffic.

AES70 does not provide a complete device implementation model. AES70 models the control and monitoring functions of a device, not its entire signal path. If a particular device element has no remotely controllable features, then that element need not be represented in the device's AES70 protocol interface.

0.2. Architectural goals and constraints

AES70 is based upon the following features and requirements:

Functionality

AES70 supports the following functions:

1. Discover the AES70 devices that are connected to the network.
2. Define and undefine media stream paths between devices.
3. Control operating and configuration parameters of an AES70 device.
4. Monitor operating and configuration parameters of an AES70 device.
5. For devices with reconfigurable signal processing and/or control capabilities, define and manage configuration parameters.
6. Upgrade software and firmware of controlled devices. Include features for fail-safe upgrades.

Security

AES70 supports the following security measures for control and monitoring data:

1. Entity authentication
2. Prevention of eavesdropping
3. Integrity protection
4. Freshness - *"Freshness" in this context means certainty that replayed messages in a replay attack on a protocol will be detected as such.*

Scalability

AES70 supports networks with up to at least 10,000 application devices. AES70 imposes minimal restriction on the physical distribution of application devices.

Availability

AES70 supports high availability by offering:

1. Device supervision of AES70 devices.
2. Supervision of network connections to AES70 devices.
3. Efficient network re-initialization following errors and configuration changes.

Robustness

AES70 supports robustness by offering:

1. A mechanism for operation confirmation.
2. A mechanism for handling loss of control data.
3. A mechanism for handling device failure of AES70 devices.
4. Recommendations on network robustness mechanisms that network implementers may use.

Safety compliance

AES70 allows implementations of media networks that conform to life-safety emergency standards.

Compatibility

As AES70 evolves, it will maximize compatibility among its different versions. A controller based on one version of AES70 operates with a device based on another version of AES70 in the following manner:

1. For a device based on an older version of AES70, the controller which is based on a newer version will function as if it were based on the same version of AES70 as the device.
2. For a device based on a newer version of AES70, the controller which is based on an older version will be able to control and monitor all the functions of the device defined in the controller's version of AES70, and will not interfere with functions defined only in the device's version of AES70.

Analyzability

AES70 defines diagnostic functions that allow access to the following information:

1. Version information of all components, hardware and software, of each device
2. Network parameters of a device - for example, MAC address, IP address
3. Device status (including status of devices' network interfaces)
4. Media stream parameters (for each active receive and/or transmit media stream of a device)

0.3. Document conventions

In what follows, the phrase "AES70 supports 'x' ", where 'x' is some function or feature should be interpreted to mean that AES70 defines one or more mechanisms by which a device will be able to implement feature 'x' in an AES70-compliant manner.

Numerical values are decimal unless otherwise stated.

A **Courier** typeface is used to identify **programmatic names** to distinguish them from regular text.

Where a term is first introduced in body text, the term will be set in an *italic* typeface.

When normative references are cited in the text they are [enclosed in brackets].

1. Scope

AES70 defines a scalable control-protocol architecture for the control and monitoring of professional media networks. AES70 addresses device control and monitoring only; it does not define standards for transporting streaming media or for describing media content.

This Part 1 describes the models and mechanisms of the AES70 Open Control Architecture. These models and mechanisms together form the AES70 Framework. This document should be read in conjunction with AES70-2: Class Structure, and AES70-3: OCP.1 Protocol for IP Networks.

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.

AES17. *AES standard method for digital audio engineering - Measurement of digital audio equipment*, Audio Engineering Society, New York, NY., US.

AES70-2. *AES standard for audio applications of Networks - Open Control Architecture - Part 2: Class structure*, Audio Engineering Society, New York, NY., US.

AES70-3. *AES standard for audio applications of Networks - Open Control Architecture - Part 3: Protocol for TCP/IP Networks*, Audio Engineering Society, New York, NY., US.

IEEE-1588. *1588-2008 - IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems*, Institute of Electrical and Electronic Engineers (IEEE), Piscataway, New Jersey, US.

ISO-9787. *ISO 9787:2013 - Robots and robotic devices -- Coordinate systems and motion nomenclatures*. International Standards Organization (ISO), Geneva, Switzerland.

ISO/IEC-10646-1. *Information technology - Universal Multiple-Octet Coded Character Set (UCS) - Part 1: Architecture and basic multilingual plane*. International Standards Organization (ISO), Geneva, Switzerland.

ITU-1(8). *ITU-R BS.2076.1, Audio Definition Model; chapter 8, Coordinate System*. International Telecommunications Union, Geneva, Switzerland, 2017 June.

SMPTE-2059-2. *SMPTE Profile for Use of IEEE-1588 Precision Time Protocol in Professional Broadcast Applications.* Society of Motion Picture and Television Engineers, Piscataway, NJ, US.

WGS-84. US Department of Defense World Geodetic System 1984. Third edition, NIMA TR8350.2. National Imagery and Mapping Agency, Washington, DC, 2000 January 3.
<http://earth-info.nga.mil/GandG/publications/tr8350.2/wgs84fin.pdf>.

3. Terms, definitions and abbreviations

For the purposes of this document, the following terms and definitions apply.

3.1. AES70-compliant interface

AES70 interface

a network interface compatible with a protocol defined in compliance with this standard.

3.2. Media device

network-connected equipment that originates or accepts media signals and exposes an AES70-compliant interface to controllers on the network to allow control and/or monitoring of its functions.

3.3. Non-media device

network-connected equipment that does not originate or accept media signals, but exposes an AES70-compliant interface to controllers on the network to allow control and/or monitoring of non-media functions.

3.4. AES70 device

Device

a media device or a non-media device. Where the context is clear, the shorter version of the term is used.

3.5. Non-AES70 device

network-connected equipment that does not expose an AES70-compliant interface.

3.6. Device model

the control model for the device as seen by AES70.

3.7. Controller

network-connected software element whose function is to control and/or monitor devices via an AES70-compliant protocol interface. A controller may be hosted in a dedicated computer, or may be a software element running in a device or in some other environment.

3.8. Control

"control" should generally be interpreted to mean "control and monitoring".

3.9. Control protocol

an application protocol whose purpose is the remote control and monitoring of the application functions of devices attached to a network.

3.10. Media stream

a digital data stream that carries a sampled digital audio or video signal over a network or over a point-to-point connection.

**3.11. Open Control Protocol
OCP**

a network protocol defined in accordance with the AES70 specification

3.12. OCP.1

protocol that implements AES70 for TCP/IP networks. Standardized in AES70-3.

**3.13. Protocol data unit
PDU**

in a layered system, a unit of data that is specified in a protocol of a given layer and contains data relevant to the operation of that layer.

**3.14. Object Number
ONo**

an arbitrary 32-bit integer used to identify an AES70 control object. Object Numbers are unique within each given device.

**3.15. Precision Time Protocol (PTP)
the time protocol defined by [IEEE-1588].****3.16. Program or AES70 Program**

an execution specification that is run within a device by an AES70 Task

**3.17. Task
AES70 Task**

a process that runs an AES70 Program within a device

**3.18. Voltage-controlled amplifier
VCA**

an analogue electronic amplifier whose gain is controlled by an external direct-current control voltage input.

**3.19. Binary Large Object
BLOB**

a contiguous block of binary data whose format and, in some cases, length are outside the scope of this standard.

3.20. Robustness

the ability of a device to cope with errors during operation despite abnormalities in signal, control input, network operation, or operating environment.

3.21. Availability

the proportion of time a device is in a functioning condition.

3.22. Alignment level

a defined anchor point that represents a reasonable or typical level. See [Wiki-001].

4. Top level design**4.1. General**

AES70 supports control and monitoring of AES70 devices at the application level. AES70 does not perform audio or video stream transport, but is designed to integrate with various audio and video signal transport schemes.

AES70 protocol is extensible, to allow the orderly incorporation of new device types and device upgrades, and generally to support upwards-compatible evolution of function in media networks.

AES70 is intended to support practical media networks that are easy to set up and operate. For simple networks of 100 nodes or fewer using recommended switches, the setup procedure should not require technical staff to have advanced networking knowledge. To this end:

1. AES70 networks can operate using industry-standard data network equipment.
2. AES70 devices can coexist harmlessly with non-AES70 devices.
3. AES70 networks may operate in a secure or unsecure mode, as products and applications require.

4.2. Object orientation

4.2.1. General

AES70 describes the control interface of a communicating device as a set of *objects*. Each object is a software element that is an instantiation of a specific *class*, and has a *state* (that is, a set of data elements called *properties*) and a set of procedural actions (called *methods*) defined by that class.

All actions and features of AES70 protocols are defined in terms of classes. The functional scope of a protocol is equivalent to the functional repertoire of the classes it implements. The set of classes fully determines what types of objects may be instantiated in the communicating devices.

A protocol defined in this way is termed an *object-oriented protocol*, and is defined by the union of the following four sets:

1. Class definitions, which define the types of objects that may exist within devices.
2. Naming and addressing rules, which define how the objects and their attributes are identified.
3. *Protocol data unit (PDU)* formats, which specify the actual formats of transmitted and received data.
4. Protocol data unit (PDU) exchange rules, which define the communication sequences used to effect information exchange.

NOTE. This specification is expressed in object-oriented design terms. However, this does not imply that implementations of those protocols must be programmed in an object-oriented style. The choice of object-based or non-object-based implementation is up to the implementer.

4.2.2. Classes

4.2.2.1. Content

Every AES70 class shall consist of the following:

1. A set of elements (properties, methods, and events - see clause 4.2.2.7)
2. A unique *class identifier*.
3. Exactly one parent class (except for the root class, which has no parent).

NOTE. Standard class names all begin with "**Oca**" (see [AES70-2]).

4.2.2.2. Inheritance

AES70 classes shall be defined as nodes in a hierarchical tree-like structure which shall start with single, elemental node (the *root class*), and shall be arranged in order by inheritance. Inheritance means that each class is considered to be a more specialized entity (the *child class*) derived from another, less specialized class (the *parent class*). A class shall exhibit (*inherit*) all the features of its parent, except where the definition of that class specifically overrides an inherited feature.

4.2.2.3. Nonstandard classes

Over the lifetime of this standard, it will often occur that specific products, especially complex ones, will require specialized control classes that are not appropriate for inclusion in the standard class tree. The standard allows for this through four policies:

1. A nonstandard class may inherit from any standard class, or from another nonstandard class.
2. A nonstandard class should be attached to the standard class tree at the most specific level that is appropriate.
3. Identifiers of nonstandard classes shall be appropriately constructed using *authority keys* - see clause 4.2.2.6, and clause 4.2.2.6.2.
4. Nonstandard classes shall be defined in accordance with the inheritance rules listed in clause 4.2.2.8.

4.2.2.4. Containment

In this specification, a class will sometimes be described as *containing* another class. This means that an object of the given class incorporates objects of the other class.

If the containing class object is deleted, the objects of the classes it contains shall be deleted.

4.2.2.5. Collection

In this specification, a class will sometimes be described as *collecting* another class. This means that an object of the given class incorporates *references* to the objects of the other class.

If the collecting class object is deleted, the objects of the classes it collects shall not be deleted.

4.2.2.6. Class identifiers

A class identifier (*class ID*) shall consist of a *lineage key* and a *version number*. Every class ID includes a version number to uniquely identify the revision level of the class.

In the following sections, a class ID is represented as $\{n; i_1 \cdot i_2 \cdot i_3 \dots\}$, where n is the class version number, and $i_1 \cdot i_2 \cdot i_3 \dots$ is the lineage key, as described below.

4.2.2.6.1. Lineage keys

Each class shall be identified by a hierarchical key nominally of the form $i_1 \cdot i_2 \cdot i_3 \dots$ where $i_1 \cdot i_2 \cdot i_3 \dots$ shall be a positive nonzero integer values that uniquely identify a class within its siblings at a particular level of the class tree. i_1, i_2, i_3 and so on are referred to as *class indices*. Class index values may be non-consecutive.

Any class index other than the first (i_1) may optionally be preceded by an *authority identifier*. An authority identifier is a special binary construct that identifies the defining authority of the index values following. If no

authority identifier appears in a lineage key, the identified class is a standard one whose authority is this AES70 standard. Authority identifiers are discussed further in 4.2.2.6.4 below.

The lineage key of each class shall consist of a set of class indices, possibly prefixed by authority identifiers, which identifies the entire lineage of the class, beginning from the root class, extending down through all related child classes, and ending at the class in question. The key may contain as many class indices as needed to describe the layers of the hierarchy.

For example, for a standard class *X* whose lineage key is **1•2•12•7**, the lineage key shall be interpreted left-to-right as follows:

- **1** designates the root class.
- **1•2** designates a child of the root class.
- **1•2•12** designates a child of the class whose parent is **1•2**.
- **1•2•12•7** designates class *X*, a child of the class whose parent is **1•2•12**.

4.2.2.6.2. Standard class IDs

Standard class IDs shall be allocated in AES70-2, which will be revised from time to time to include new classes and new versions of existing classes.

4.2.2.6.3. Nonstandard class IDs

Nonstandard class IDs may be created by the manufacturer of a device or by any other organization that develops class definitions. The source of a class ID is called the class's *authority*.

A nonstandard class ID shall be constructed by prefixing the nonstandard portion of the lineage key with an authority identifier. The authority identifier shall be inserted in the lineage key immediately before the index of the first nonstandard class. In the following details, authority keys will be indicated by "A", as in {*n*; *i*₁•*i*₂•**A**•*i*₃ ...}.

The interpretation of a lineage key that includes an authority identifier A shall be as follows:

1. Every index to the left of A shall identify a standard AES70 class;
2. Every index to the right of A shall identify a nonstandard class defined by the authority identified by an authority key (see clause 4.2.2.6.5); and
3. The index immediately to the left of **A** shall identify a class is the parent of the class whose index is immediately to the right of **A**.

For example, the lineage key of {**1•1•2•A•5**} identifies a nonstandard class whose parent is {**1•1•2**}, whose authority is **A**, and whose own (nonstandard) class index is 5.

All class indices to the right of the authority index shall be considered to be nonstandard. For example, the lineage key {**1•1•2•A•5•4**} identifies a nonstandard class with index 4 that is a child of the nonstandard class with index 5 that is a child of the standard class {**1•1•2**}.

NOTE. This rule is a corollary of the rule that a standard class shall not inherit from a proprietary class.

4.2.2.6.4. Lineage key format

The normative specification of the lineage key format is given here.

1. Each index value i_k shall be a 16-bit unsigned integer.
2. To form the lineage key, index values shall be concatenated left to right in order of inheritance.
3. An authority key (see clause 4.2.2.6.5) shall be 48 bits long and may be inserted before any index value except the leftmost (root).
4. The leftmost index value shall be the index of the OCA root class, which always shall have the value 1.

Class index values may be freely assigned subject to the following rules:

1. The value 0 shall not be used.
2. Values 1...32767 ($0001_{16}...7FFF_{16}$) may be used for standard or nonstandard class indices
3. Values 32768...65279 ($8000_{16}...FEFF_{16}$) shall be reserved for legacy AES70-2015 nonstandard classes.
4. Values 65280...65534 ($FF00_{16}...FFFE_{16}$) shall be volatile and intended for testing.
5. Value 65535 ($FFFF_{16}$) shall be a special flag identifying the start of an authority identifier.

The use of values in the range 32768...65534 is **deprecated**.

4.2.2.6.5. Authority key

The format of an authority key shall be as follows (left to right, hexadecimal notation):

Bits 0-15	$FFFF_{16}$
Bits 16-23	00_{16}
Bits 24-48	Authority's IEEE Public Company ID (Public CID) or Organizational Unique Identifier (OUI)

NOTE: The IEEE (Institute of Electrical and Electronic Engineers) Public CID is a 24-bit identifier which uniquely identifies an organization. Any incorporated organization may receive a unique Public CID from the IEEE upon request and payment of a one-time fee. See IEEE-1 and IEEE-2. CIDs and OUIs are non-overlapping, so either will serve to identify the authority.

4.2.2.7. Methods, properties, and events

AES70 classes shall have elements which allow access to their data and operating states.

The elements of AES70 classes shall be as follows:

Properties	A class shall define properties, which are variables that store the class's specific control or monitoring parameters that are accessible from the control network.
Methods	A class shall define methods, which are procedures that controllers may invoke via AES70 protocol commands to retrieve and change property values, to change class operating states, and to perform other actions.

Events A class may define one or more events, which are callbacks initiated by devices to inform controllers of specific occurrences. To receive events, controllers shall first *subscribe* to them. See clause 6.

These elements shall be used to define protocol exchanges between devices and controllers. The manner in which class elements shall be represented in communications-protocol elements is specific to each AES70 protocol. For one example, see [AES70-3].

4.2.2.7.1. Element IDs

In the class tree, every method, property, and event shall be assigned not only a name, but also an *element ID* of the form: **LLtNN**, where

LL shall be the two-digit level of the class tree at which the class is defined.

For example, the global base class **OcaRoot** is defined at level **01**. Children of **OcaRoot** will be defined at level **02**. Grandchildren will be defined at level **03**. And so on.

t shall be a type code: **p** for properties, **m** for methods, **e** for events.

NN shall be a sequence number starting at **01** for each type within each class.

Within each class, element ID values shall be unique.

EXAMPLE 1

01p01 is the first property of a class defined at tree level **01**. In this case, **OcaRoot** is the only class defined at level **01**, so **01P01** is the first property of **OcaRoot**.

EXAMPLE 2

03m02 is the second method of a class defined at tree level **03**. There are several classes defined at level **03** - this ID would apply to the second method of any of these classes.

NOTE 1. Element ID rules are intended to provide a means for uniquely identifying all the native and inherited methods in any given class, and to allow for future expansion of the tree at any level without duplicating identifiers.

NOTE 2. For an example of this approach, see the class **OcaGain** in annex A.

NOTE 3. An element ID may be a property ID, a method ID, or an event ID, depending on the type of element it identifies.

4.2.2.7.2. Method status

All methods shall return a status code which indicates successful completion or, if unsuccessful, the general reason for failure. Status code values are described by the **OcaStatus** datatype defined normatively in [AES70-2].

4.2.2.7.3. Protocol invariance

For any given class, the following items shall be constant, regardless of which AES70 protocol is used:

1. The set of properties, methods, and events; and
2. The set of element IDs.

4.2.2.7.4. Text format

All text in AES70 properties, method parameters, and event parameters shall be in UTF-8 format. See [ISO/IEC 10646-1].

4.2.2.8. Inheritance and updating rules

Inheritance rules ensure that, by creating new classes from existing classes, new features can be added to the protocol in a manner that does not impact the operation of existing products or systems. Inheritance ensures that new child classes will support, at a minimum, the functions and interfaces of their parents.

In AES70, the rules for inheritance are:

1. Any given class except the root class shall inherit from exactly one other class.
2. A class shall implement all the properties, methods, and events of its parent class.
3. A child class may expand its parent's definition by:
 - a. Adding new properties, methods, and/or events; and/or
 - b. Enhancing the definitions of existing properties, methods, and/or events, In this case, the enhanced definitions shall support all functions defined by the parent class.
4. A child class's inherited methods and events shall retain the respective element IDs of its parent class.
5. A standard class shall not inherit from a proprietary class.

When updating an existing class, the following rules shall apply:

1. The class version number shall be incremented.
2. The updated class shall implement all the properties, methods, and events of the existing class.
3. The updated class may expand the existing class's definition by:
 - Adding new properties, methods, and/or events; and/or
 - Enhancing the definitions of existing properties, methods, and/or events, In this case, the enhanced definitions shall support all functions defined by the existing class.
4. An updated class's methods and events shall retain the respective element IDs of its parent class.

4.2.3. Instantiation of classes

As required to create its network control interface, a device shall instantiate classes as objects. Each such object shall be identified by a unique *Object Number (ONo)*.

4.3. Messages

4.3.1. Basic mechanism

Control and monitoring operations shall be implemented by messages in protocol data units (PDUs) that pass between one object and another object which may be in a different device. An AES70 message shall be of one of the following three types:

Command	request devices to return data and/or perform actions.
Acknowledgement	report success or failure of a command, and return requested data, if any.
Notification	report the occurrence of certain events within devices, and provide related data.

With the exception of the device reset message (clause 16), every AES70 command message shall be acknowledged by a corresponding acknowledgement message. Notification messages shall not be acknowledged.

4.3.2. Message delivery services

AES70 recognizes two services for message delivery: *Reliable* and *Fast*.

The Reliable delivery service shall use an assured means of transport that the network offers. The Reliable service shall not be used for multicasting. For example, in TCP/IP networks, the reliable service uses TCP.

The Fast delivery service may use a lower-overhead, lower-reliability means of transport. The Fast service may be used for multicasting, if the network supports it. For example, in TCP/IP networks, the fast service uses UDP.

All AES70 command and acknowledgement messages shall be transmitted using the Reliable delivery service. Notification messages may use either service. A Fast delivery service may be used by the subscription mechanism, described in clause 6 , and shall be used by the device reset mechanism, described in clause 16.

NOTE: For some types of networks, Reliable and Fast services may be identical.

5. Device model

5.1. Device configurability

The configurability of an AES70 device may be *fixed*, *pluggable*, *partially-configurable*, or *fully-configurable* as shown below in Table 1.

Fixed and pluggable devices are termed *static* devices, because controllers cannot change their configurations. Partially-configurable and fully-configurable devices are termed *dynamic* devices, because their configurations can be varied while online.

Details of configuration management are in clause 5.4.3 and AES70-2. Creation and deletion of objects are discussed further in clause 5.10.

Table 1 - Configurability

Configurability	Type	Description
Fixed	static	The device has a permanently assigned object repertoire and signal-flow topology, defined at time of firmware programming.
Pluggable	static	The object repertoire and signal-flow topology of the device may be changed while the device is offline, by plugging and unplugging of hardware modules, adjustment of physical controls, reloading or readjustment of software, or other manual means.
Partially-configurable	dynamic	Controllers may change the signal-flow topology of the device while online.
Fully-configurable	dynamic	A superset of 'partially-configurable', with the addition that controllers may create and delete objects inside the device while online.

5.2. Object addressing

Within an AES70 device, each object (that is, each instantiation of a specific class) shall be identified uniquely by an *object number (ONo)*. Depending on device configurability, ONos shall be assigned at different times, as shown in Table 2.

NOTE: In specific implementations, developers may use specific object numbering schemes to optimize device performance and/or to ease object management. For example, a developer might choose a device's object number values to correspond with internal table index values, in order to facilitate rapid decoding of incoming commands.

Table 2 - ONo assignment

Configurability	Time of ONo assignment
Fixed	time of manufacture
Pluggable	device setup time
Partially-configurable	time of manufacture
Fully-configurable	time of object creation

In fully-configurable devices, ONo values shall not be re-used when objects are deleted and recreated without an intervening device reset. In the unlikely event that the ONo address space becomes exhausted, ONo values may be re-used in any way that does not violate rules (1)-(5) below.

Device implementers may choose ONo values as desired, subject to the following rules:

1. Each Object Number (ONo) shall be a 32-bit integer.
2. Every object shall have a unique ONo.
3. The ONo value of zero shall not be used.
4. ONo values 01 through 4095 shall be reserved for managers (clause 5.7) and other objects that require predefined object numbers. They shall be reserved for AES70 standardization.
5. Different ONos may not refer to the same object.

NOTE. 1. Unlike some other media control protocols, ONos are not multi-field (*i.j.k. ...*) values based on some hierarchy. Using a simple 32-bit ONo value is a design choice intended to maximize protocol efficiency and to minimize device overheads in resolving object references, handling command responses, and managing errors.