

STANDARDS AND INFORMATION DOCUMENTS

AES24-2tu-1999
(withdrawn 2004)



STANDARDS

**DRAFT AES standard for sound system control -
Application protocol for controlling and
monitoring audio devices via digital data
networks -**

**Part 2: Data types, constants, and class
structure**

(Publication for trial use only)

(Withdrawn 2004)

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.
60 East 42nd Street, New York, New York 10165, USA



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.



This list is current as of 2017/7/25

STANDARDS AND INFORMATION DOCUMENTS

PROPOSED DRAFT for trial use and discussion AES standard for sound system control — application protocol for controlling and monitoring audio devices via digital data networks — Part 2, data types, constants, and class structure

This document was developed by a writing group of the Audio Engineering Society Standards Committee (AESSC) and has been prepared for discussion according to AES policies and procedures. It has been brought to the attention of International Electrotechnical Commission Technical Committee 100. Existing international standards relating to the subject of this document were used and referenced throughout its development.

AESSC wishes to provide opportunity for discussion and trial use before the document is presented as a call for comment leading to finalization of the standard. It is therefore initiating a process of public discussion via the Internet. The AESSC recognizes that a configuration standard such as this document will ultimately be tested in its application and that trial use will show need for additional changes, modifications and additions. As initially made available, the standard lacks basic components. Primary among them are the network-management classes and the matrix classes. Without the set of network-management classes (which include the events and methods needed for system initialization), an AES-24 system cannot start.

This document is intended to change and will be prepared as a call for comment only after the working group is satisfied that the document is ready. Changes will be made and the document updated as they are agreed upon, based on the discussion, by a task group of SC-10-02. As editorial changes, particularly in the organization of the document, become necessary they will be made by the AESSC secretariat. The working group will be informed by reflector mail each time the document posting is updated. Participants in the discussion should be certain they are working with the latest posting as shown in the headers and footers of the document.

Address discussion to SC_10_02@aessc.aes.org. **Only discussion so addressed will be considered.** Discussion must be restricted to this document only, except to the extent that this document is affected by other documents related to it. Send unrelated discussion about other documents separately. Persons who wish also to become members of the working group should so indicate and include all address information as shown at <http://www.aessc.org/aessc/memb.htm>.

Recipients of this document are invited to submit, with their discussion, notification of any relevant patent rights of which they are aware and to provide supporting documentation.

Because this document is a proposed draft and is subject to change, no portion of it shall be quoted in any publication without the expressed permission of the AES, and all published references to it must include a prominent warning that the proposed draft will be changed and must not be used as a standard.

PROPOSED DRAFT AES24-2-TU

PROPOSED DRAFT AES standard for sound system control — application protocol for controlling and monitoring audio devices via digital data networks — Part 2, data types, constants, and class structure

Published by

Audio Engineering Society, Inc.

Copyright © 1999 by the Audio Engineering Society

Abstract

This standard specifies the class hierarchy for the AES-24 application protocol. AES-24 classes make up the conceptual framework from which AES-24 objects and the AES-24 application protocol and its message definitions are derived. This standard is intended to be used by developers of software and firmware whose applications and/or devices will be used in conjunction with an AES-24 network.

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. 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.

Contents

Foreword.....	5
1 Scope.....	6
2 Normative References.....	6
3 Definitions.....	6



3.1 Classes.....	6
3.2 Device states	7
4 Security	7
5 Gateways.....	7
6 Class member identification.....	7
6.1 Class member.....	7
6.2 Class level	7
6.3 Member IDs	8
6.4 Member ID notation.....	8
6.5 Resolving member ID ambiguity	10
7 Data-type reference	10
7.1 Data-type descriptions.....	11
8 Status code reference	13
9 Class reference	13
9.1 Class hierarchy.....	15
9.2 Root.....	16
9.3 Network Management.....	22
9.4 Registry Manager.....	23
9.5 Device Manager	24
9.6 Functional	27
9.8 Actuator	27
9.8 Z-Domain Filter	28
9.10 Classical Filter.....	29
9.11 Parametric EQ.....	31
9.12 Gain Control.....	31
9.13 Generator.....	32
9.14 Delay	32
9.15 Scalar	33
9.16 Switch	34
9.17 Duck/Mute	34
9.18 Ratio.....	35
9.19 Threshold	36
9.20 Attack/Release	37
9.21 Sensor.....	38
9.22 Detector.....	39
9.23 Continuous	39
9.24 Synchronized Continuous	40
9.25 Exception	41
9.26 User Interface.....	42
9.27 Selector	42
9.28 Menu Control.....	43
9.29 Discrete Indicator.....	44
9.30 Text Input/Output.....	44
9.31 Numeric Input/Output	45
9.32 Spinwheel.....	46
9.33 Continuous Control	47
9.34 Meter.....	47
9.35 2-D Curve.....	48
9.36 Intermediate	49
9.37 Combiner/Splitter.....	50
9.38 Slewer	50
9.39 Data Translator.....	51

PROPOSED DRAFT FOR TRIAL USE AND DISCUSSION ONLY

secretariat 1999-02-27 PROPOSED DRAFT AES24-2-TU 99/02/2818:41

9.40 Matrix Translator	51
10 Matrix Classes.....	52
11 Proposals for software extensions and enhancements.....	52



Foreword

[This foreword is not part of PROPOSED DRAFT AES *standard for sound system control — application protocol for controlling and monitoring audio devices via digital data networks — Part 2, data types, constants, and class structure*, PROPOSED DRAFT AES24-2-tu.]

PROPOSED DRAFT AES standard application protocol for controlling and monitoring audio devices via digital data networks — Data types, constants, and class structure —

1 Scope

This document describes specific classes within the class hierarchy for the AES-24 application protocol. In order to aid in orderly modification and extension of the protocol, this standard models AES-24 through an object-oriented framework in which one class may inherit characteristics from another.

To achieve a conceptual understanding of AES-24, it is not necessary to understand the details of each class as described in this (Part 2) document. However, the details described herein must be understood and followed in order to program applications and devices which will interact on an AES-24 network.

2 Normative References

The following standard contains provisions that, through reference in this text, constitute provisions of this document. At the time of publication, the edition indicated was valid. All standards are subject to revision, and parties to agreements based on this document are encouraged to investigate the possibility of applying the most recent editions of the indicated standards.

1) AES24-1-1999 *AES standard for sound system control — Application protocol for controlling and monitoring audio devices via digital data networks — Part 1: Principles, formats, and basic procedures*. New York, Audio Engineering Society, 1999.

3 Definitions

3.1 Classes

3.1.1

descendant class

from the perspective of a particular class, class that has been derived either directly, or indirectly from that class

3.1.2

leaf class

class that has no descendants

3.1.3

root class

single class that descends from no class and from which all other classes directly or indirectly descend

3.1.4**branch**

in the context of a class hierarchy, inheritance path between any non-root class and the root

3.1.5**member**

property, method, or event of a class

3.2 Device states**3.2.1****on-line**

state of an AES-24 device that has confirmed its address and is participating in normal AES-24 internetworking transactions

3.2.2**off-line**

state of a functional AES-24 device that is either powered on or in stand-by, but has not confirmed its address and is not participating in normal AES-24 internetworking transactions

4 Security

Under consideration. See clause 11.

NOTE – Network-level security shall be the responsibility of the network. AES-24 assumes that non-designated recipients of a message shall not be able to decrypt an AES-24 message. `lock_object()`, for example, returns a key in the form of an `AESSTRING[]` that must be returned if the object is to be unlocked. AES-24 assumes that other objects should not be able to see this string as it passes through the network.

5 Gateways

AES-24 gateways and subnetwork name assignments are under consideration. See clause 11.

6 Class member identification**6.1 Class member**

“Class member” is a generic term that shall denote either a property, method, or event of a class.

6.2 Class level

The level of a class identifies how many ancestor classes it has inherited — in other words, the number of classes removed from the root where the class is defined. For example, if class *x* inherits from class *y*, and class *y* inherits from the root class, then class *x* is two classes removed from the root class, and thus has a class level of 2.

Therefore, properties assigned in the root shall have a class level identifier equal to 0; properties assigned in classes which are immediate children of the root shall have a class level identifier equal to 1; properties assigned in classes which are two classes descended from the root shall have a class level identifier equal to 2; and so on. The maximum value of the class level identifier shall be F_{16} .

6.3 Member IDs

A member ID is a unique 14-bit number made up of a 4-bit class level identifier concatenated to a 10-bit index to a member within its class. Properties, methods, and events shall be identified by member IDs that identify the level of the class where the member was defined, and the index of the member within its defining class. Therefore, the maximum value of a member index shall be $3FF_{16}$.

Note that property, method, and event indices shall be independently assigned within each class.

6.4 Member ID notation

For convenience, member IDs shall be written as CpI, CmI, or CeI to denote property, method, and event IDs, respectively, where C represents, in hexadecimal, the zero-based class level where the member was defined, and where I represents, in hexadecimal, the zero-based index of the member within its class. For example, 3m4 denotes the 5th method of the 3rd descendant class of the root (within the branch of the class in question). The root shall have no ancestors, and thus shall be the only class with a class level of 0.

Figure 1 shows a sample class hierarchy that demonstrates the member identification scheme. Notice, however, that there is no relational pattern to the class identification scheme since there is no particular pattern to the way new classes are added to the AES-24 class hierarchy.

NOTE – This figure does not inductate the actual class hierarchy of AES-24.

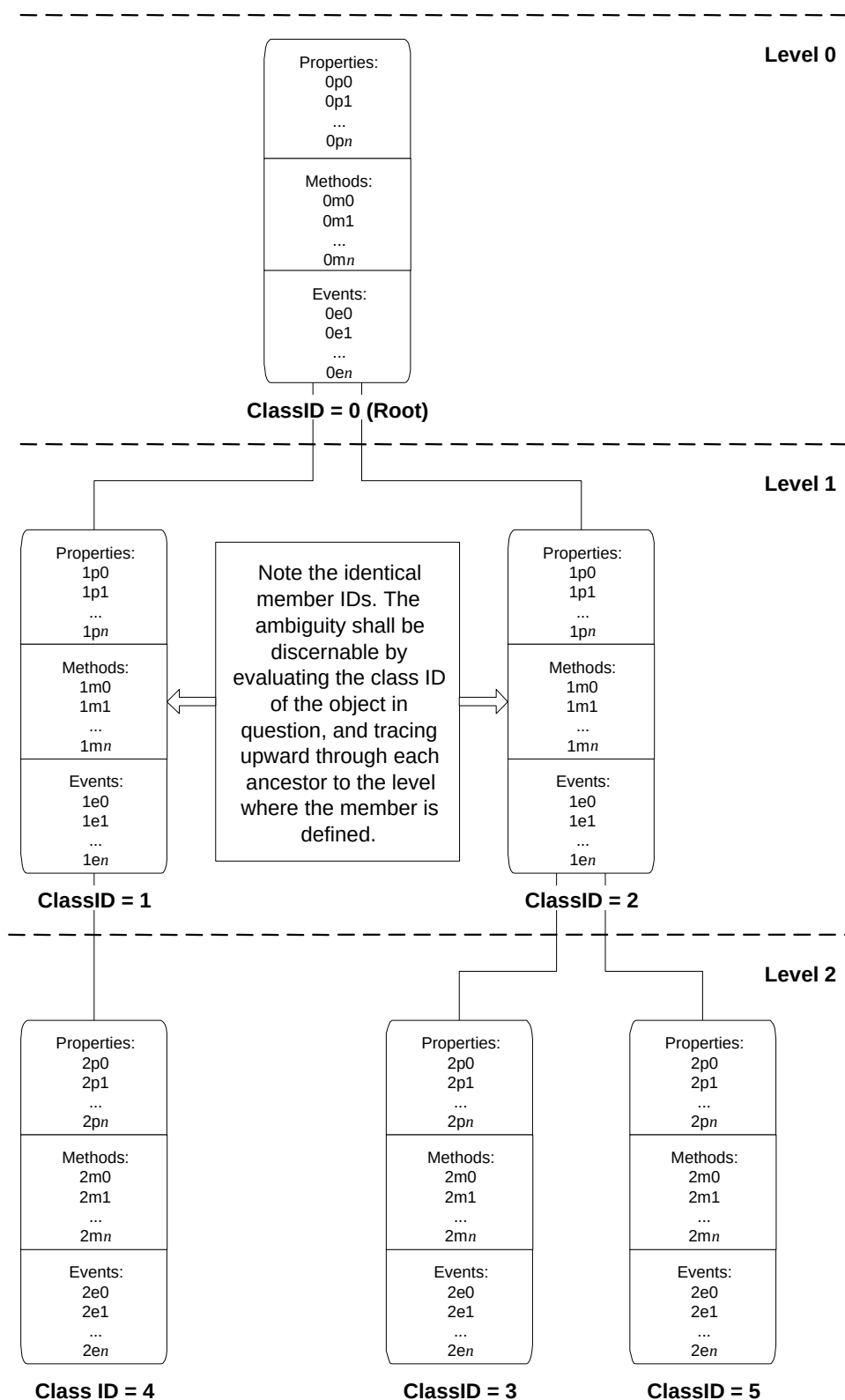


Figure 1 — Sample class hierarchy

6.5 Resolving member ID ambiguity

No ambiguity shall arise among the potentially numerous uses of the same member ID in the class hierarchy. For example, there are two classes with member IDs of 1m0. Assuming, as in Figure 1, that we have an object with a ClassID of 3, its inherited method, identified as 1m0, can be found by first determining the level of the class (i.e., level 2), and then by tracing up through each ancestor to the level indicated by the method ID (i.e., level 1). The ancestor class that is reached at the indicated level is the one that holds the method 1m0 that has been inherited by the object in question.

7 Data-type reference

Table 1 — AES-24 data types

Data type	ANSI C-Style Typedef
UBYTE	typedef unsigned char UBYTE; //8-bit unsigned scalar
UWORD	typedef unsigned short int UWORD; //16-bit unsigned scalar
UDWORD	typedef unsigned long int UDWORD; //32-bit unsigned scalar
UQWORD	typedef unsigned __int64 UQWORD; //64-bit unsigned scalar
SBYTE	typedef signed char SBYTE; //8-bit signed scalar
SWORD	typedef signed short int SWORD; //16-bit signed scalar
SDWORD	typedef signed long int SDWORD; //32-bit signed scalar
SQWORD	typedef signed __int64 SQWORD; //64-bit signed scalar
FLOAT	typedef float FLOAT; //32-bit floating point
DOUBLE	typedef double DOUBLE; //64-bit floating point
AESDB	typedef FLOAT AESDB;
AESHERTZ	typedef FLOAT AESHERTZ;
AESSTRING	typedef UBYTE AESSTRING[256]; //Fixed-length string
AESSV8	typedef SBYTE AESSV8[];
AESSV16	typedef SWORD AESSV16[];
AESUV16	typedef UWORD AESUV16[];
AESSV32	typedef SDWORD AESSV32[];
AESUV32	typedef UDWORD AESUV32[];
AESSV64	typedef SQWORD AESSV64[];
AESUV64	typedef UQWORD AESUV64[];
AESCLASSID	typedef UWORD AESCLASSID;
AESOBJHANDLE	typedef UWORD AESOBJHANDLE;
AESDEVHANDLE	typedef UWORD AESDEVHANDLE;
AESNETHANDLE	typedef UWORD AESNETHANDLE;
AESOBJECTADDR	typedef struct { AESNETHANDLE net_handle; //Most significant word shown 1 st AESDEVHANDLE dev_handle; AESOBJHANDLE obj_handle //Least significant word last }AESOBJECTADDR;
AESDEVICEADDR	typedef struct { AESNETHANDLE net_handle; AESDEVHANDLE dev_handle; }AESDEVICEADDR;
AESBOOLEAN	typedef BYTE AESBOOLEAN; enum {FALSE, TRUE};
AESMEMBERID	typedef struct { UWORD index : 10; //Note: least sig. bits shown 1 st UWORD level : 4; //bits 0..9 }AESPROPERTYID; //bits 10..13
AESPADDEDMEMBERID	typedef struct { UWORD index : 10; //bits 0..9 UWORD level : 4; //bits 10..13 UWORD : 2; //pad bits 14..15 }AESPADDEDPROPERTYID;

AESMESSAGEID	typedef struct { UWORD index : 10; //bits 0..9 UWORD level : 4; //bits 10..13 UWORD type : 2; //bits 14..15 }AESMESSAGEID;
AESSTATUSCODE	typedef { UWORD value : 15; //bits 0..14 UWORD type : 1; //bit 15 }AESSTATUSCODE;
AESSENDMESSAGE	typedef struct { UBYTE AES_ver; UBYTE destination[6]; UBYTE reply[6]; AESMESSAGEID message_id; UBYTE sequence_no; UBYTE parameters[]; //Size is message dependent }AESSENDMESSAGE;
AESREPLYMESSAGE	typedef struct { BYTE version; BYTE destination[6]; BYTE reply[6]; AESMESSAGEID message_id; BYTE sequence_no; AESSTATUSCODE status_code; BYTE parameters[?]; //Size is message dependent }AESREPLYMESSAGE;

7.1 Data-type descriptions

7.1.1 SBYTE, SWORD, SDWORD, SQWORD

SBYTE, SWORD, SDWORD, and SQWORD, respectively, shall be scalar types used to denote 8-, 16-, 32-, and 64-bit signed integers.

7.1.2 UBYTE, UWORD, UDWORD, UQWORD

UBYTE, UWORD, UDWORD, and UQWORD, respectively, shall be scalar types used to denote 8-, 16-, 32-, and 64-bit unsigned integers.

7.1.3 FLOAT

FLOAT shall be a 32-bit floating point number with limits of 3.4×10^{-38} to $3.4 \times 10^{+38}$.

7.1.4 DOUBLE

DOUBLE shall be a 64-bit floating point number with limit of 1.7×10^{-308} to $1.7 \times 10^{+308}$.

7.1.5 AESDB

AESDB shall be a 32-bit floating point number.

7.1.6 AESHERTZ

AESHERTZ shall be a 32-bit floating point number.

7.1.7 AESSTRING

AES-24 strings shall be fixed-length (that is, 256 bytes) arrays of unsigned characters. They shall be NULL (ASCII 0) terminated as standard C strings.

7.1.8 AESSV8, AESSV16, AESSV32, AESSV64

AESSV8, AESSV16, AESSV32, AESSV64 shall be vectors of signed 8-, 16-, 32-, and 64-bit scalars.

7.1.9 AESUV8, AESUV16, AESUV32, AESUV64

AESUV8, AESUV16, AESUV32, AESUV64 shall be vectors of unsigned 8-, 16-, 32-, and 64-bit scalars.

7.1.10 AESCLASSID

AESCLASSID shall be the type used to define an AES-24 class ID. It shall be a 2-byte number. Class ID numbers shall be assigned sequentially in arbitrary order, starting with the root class as 0_{16} . When new classes are added to the hierarchy, their class IDs shall be assigned in sequence starting from the last class number previously assigned.

7.1.11 AESOBJHANDLE

AESOBJHANDLE shall be a 16-bit word used to uniquely represent the identity of an object within its device context.

7.1.12 AESDEVHANDLE

AESDEVHANDLE shall be a 16-bit word used to uniquely represent the identity of a device within the context of its subnetwork.

7.1.13 AESNETHANDLE

AESNETHANDLE shall be a 16-bit word used to uniquely represent the identity of a subnetwork within the context of its internetwork.

7.1.14 AESOBJHANDLE

AESOBJHANDLE shall be a type that contains the subnetwork, device, and object handles used to uniquely identify an object within an internetwork.

7.1.15 AESDEVICEADDR

AESDEVICEADDR shall be a type that contains the subnetwork and device IDs used to uniquely identify a device within an internetwork.

7.1.16 AESBOOLEAN

AESBOOLEAN, although defined as an unsigned char, shall only be assigned the values 0 and 1 to indicate false and true condition, respectively.

7.1.17 AESMEMBERID

AESMEMBERID shall be the type that defines a class member identification number.

7.1.18 AESPADDEDMEMBERID

AESMEMBERID shall be the type that defines a class member identification number. It also includes two bits (14 and 15) that are padded with zeros for ease of passing references to class members in AES-24 messages.

7.1.19 AESMESSAGEID

AESMESSAGEID shall be a type that defines the structure of the message_id field of an AES-24 message.

7.1.20 AESSTATUSCODE

AESSTATUSCODE shall be a type that defines the structure of the status_code field of an AES-24 message.

7.1.21 AESSENDMESSAGE

AESSENDMESSAGE shall be a type that defines the structure of an AES-24 send message with all the fields as defined in AES24-1.

7.1.22 AESREPLYMESSAGE

AESREPLYMESSAGE shall be a type that defines the structure of an AES-24 reply message with all the fields as defined in AES24-1.

8 Status code reference

Table 2 lists all pre-defined AES-24 generic status code values. Generic status codes shall be used when a method-specific status code either does not apply to the error, or when an error is detected external to the method to which the message was directed (such as in a front-end message dispatching function).

All AES-24 generic status codes shall be prefixed with “AGSC_” which stands for “AES-24 generic status code.”

Although the status_code of an AES-24 message is composed of two values, namely, type and value, the developer may wish to view this field as one 16-bit unsigned scalar value for ease of error checking. In that case, the value of 32768 (or 8000₁₆) shall be added to the method-specific status code values presented in the method details sub-clauses of the class reference clause.

Table 2 — Generic status codes

STATUS CODE NAME	VAL UE	DESCRIPTION
AGSC_OK	0	Indicates that method execution was successful.
AGSC_GENERROR	1	Returned to signify a general failure in processing the message. It may also be used when no other status code seems appropriate. NOTE – This status code shall be returnable for any AES-24 message since an error code may be returned before the message is delivered to the method to which it is directed.
AGSC_OBJECTLOCKED	2	Returned when write access is attempted on an object that is locked by another object.
AGSC_BADFORMAT	3	Returned when there is a syntax problem with the message. For example, when there is a missing parameter, or when message data is incorrectly formatted.
AGSC_DESTUNKNOWN	4	Destination object handle is unknown. Returned if the destination object cannot be found or does not exist.
AGSC_BADVERSION	5	Returned when the version of AES-24 designated in the AES_ver field of the message is not supported.

If a method does not supply a method specific status code for the return condition, it shall return a generic status code supplied from the preceding table. Alternatively, if a method does support method specific status codes and a return condition does not satisfy the method specific status code, then a generic status code shall be returned. For example, the method event_callback() defined in the Root class returns 0 if the event was processed by the event callback function, and 1 if it all was well but the event was simply not processed by decision. However, it may be the case that the event parameters were not formatted correctly, therefore, the returned status code should be a generic status code equal to AGSC_BADFORMAT.

In all cases when a return status code is requested, the generic status code AGSC_OK shall be returned when a method has successfully performed its defined function.

9 Class reference

The AES-24 class hierarchy is modeled using an object-oriented framework in which one class may inherit from another. A pictorial representation of the complete class hierarchy is provided in the following clause. Note that the order in which classes are represented in the diagram is not representative of the class ID number.

The remainder of the clause shows the details of each class defined by AES-24. For each class, the following data are provided:

- An inheritance path. This shows the inheritance path from the Root class to the derived class being defined. It is formatted as Root→Ancestor Classes→...→Derived Class. The inheritance path of a class can also be understood by studying the pictorial representation of the class hierarchy provided in 9.1.
- A Class ID. This shows the class identification number which is unique for each class. An object of a particular class shall initialize its class ID field to the value defined by that class.
- A general description of the class.

- Three tables providing an overview of all the properties, methods, and events that shall be provided by the class.
- Detailed information for each property.
- Detailed information for each method. Specifically, for each method, the calling and return parameters, return status codes, and important comments on general usage are provided.
- Detailed information for each event. Specifically, for each event, information on its behavior and general usage, as well as a description of each of the parameters that would be directed toward a general event callback handler are provided.

9.1 Class hierarchy

The class hierarchy shall be as shown in Figure 2.

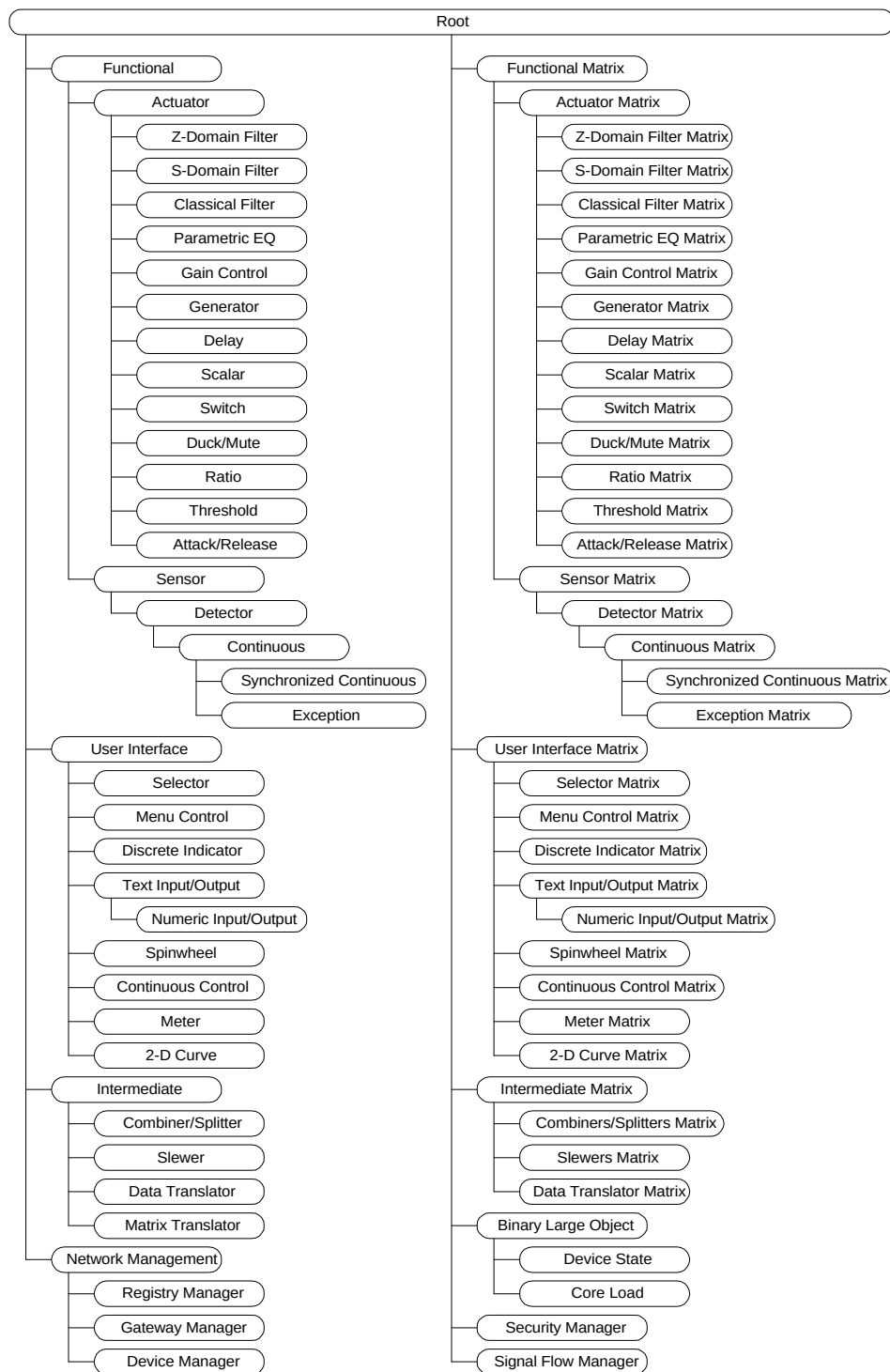


Figure 2 — AES-24 class tree