AES24-1-1999
(Revision of AES24-1-1995)

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

Published by
Audio Engineering Society, Inc.

Abstract

This standard describes the architecture of AES-24, the name assigned to an extensible application protocol for controlling and monitoring audio devices via local area networks, and, when possible in the future, devices designed for other media.

Contents

Foreword

[This foreword is not a part of *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*, AES24-1-1999.]

This document is a revision of AES24-1-1995 *AES standard for sound system control -- Application protocol for controlling and monitoring audio systems -- Part 1: Architecture.* The revision was made by the SC-10-02 Working Group on Application Protocols of the SC-10 Subcommittee on Sound System Control. At the time of finalizing the standard, SC-10-02 had 75 members from some 12 countries. This revision and the drafts of subsequent parts of the AES24 standard are based on detailed proposals prepared in late 1996 by a task group headed by J. Berryman. To enhance public participation in the development, unofficial synopses of these proposals were also prepared for the trade press by Fred Ampel.

Three additional parts of the AES24 standard are planned.

AES24-2, *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*.

AES24-3, *AES standard for sound system control — Application protocol for controlling and monitoring audio devices via digital data networks — Part 3: Transport requirements*.

AES24-4, *AES standard for sound system control — Application protocol for controlling and monitoring audio devices via digital data networks — Part 4: Internet protocol (IP) transport of AES-24.*

Many individuals contributed to the creation of this standard. At the time of its completion, the writing group that produced it included in its membership F. Ampel, D. Bavholm, J. Berryman (SC-10-02 chair), R. Caine, J. Combs, S. Crompton, K. Dalbjorn, B. Evans, K. Fitzke, N. Hamawi (writing group convenor), C. Hanna, B. Harshbarger, P. Ibbotson, M. Karagosian (SC-10-01 chair), D. Karlin, M. Lave (SC-10-02 vice-chair), R. Moses, R. Neely, J. Oakley, S. Potosky, T. Roseberry (SC-10 chairman emeritus), J. Shaettle, A. Singer, P. Smith, R. Spina, J. Stembel, L. Tyler (SC-10 chair), B. Van Der Werf, and D. Warman.

In particular, the chair would like to express his gratitude to the former chair of SC-10, Tom Roseberry, for his sure, calm guidance of the subcommittee from the time of its formation through late 1995.

Jeff Berryman
Chair, AESSC SC-10-02
1997-05-15

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

0 Introduction

0.1 Goals

The AES-24 protocol is conceived to meet the following goals.

1) Extensibility

AES-24 is extensible in a clean way, to support new equipment and concepts that may be developed.

2) Flexibility

AES-24 provides orderly methods for accommodating nonstandard devices in ways that maximize interoperability.

3) Compatibility

AES-24 supports three types of compatibility:

a) backward, in which new versions of AES-24 support products that use older versions, although not with the full functionality of the new versions;
b) forward, in which old versions of AES-24 support products that use newer versions of AES-24, although not with the full functionality of the devices;
c) lateral, in which products using custom enhancements to AES-24 interoperate with products using standard versions of AES-24, although not with the functionality of the custom enhancements.

0.2 Flexible data transmission requirements

AES-24 is conceived to operate over a variety of data transmission facilities, although AES-24 functionality may vary depending on the services provided by those facilities.

0. 3 Peer-to-peer operation

AES-24 is a peer-to-peer protocol, in which any device may initiate or accept control and monitoring commands.

0.4 Security

AES-24 supports features for implementing appropriate levels of security in control networks. Details on security are considered in AES24-2.

0.5 Conceptual model

AES-24 is based upon a conceptual model in which each piece of equipment is treated as a set of discrete functional elements. Some of these functional elements deal directly with the audio signal, while others provide management and support services.

In addition to audio processing equipment, an AES-24 internetwork should include one or more other components (often called system controllers) that provide control and monitoring functions. In the same way as for audio equipment, AES-24 models system controllers as sets of discrete functional elements.

Although it is convenient to speak of system controllers as specific components, AES-24 does not accord them any special distinction. Every product attached to an AES-24 internetwork is viewed simply as a box containing assorted functional elements. It is perfectly acceptable for a single product to perform both audio processing and system control functions.

In line with modern software engineering practice, AES-24 names the discrete functional elements of a product objects.

0.6 Architecture

The architecture of this standard is basically a four-level hierarchy. The levels are, from lowest to highest in the hierarchy:

a) the objects inside the equipment, organized into groups called devices;
b) the devices;
c) the subnetwork;
d) the internetwork.

0.7 Abstraction

This standard's object-based view of audio systems is an abstraction for control purposes and not a physical description of the systems themselves. It is appropriate to think of a product's set of standard objects as the external interface it presents to a standard internetwork, rather than its internal design. This interface represents the set of control and monitoring functions that the product's designers have chosen to make accessible via the application protocol. This set may include all of the product's functions, or only a subset of them.

0.8 Transport networks

This standard specifies the formats, rules, and meanings of audio control and monitoring commands, but makes few assumptions about the manner in which these commands will physically be transported from one object to another. The medium by which AES-24 commands move between objects is called a transport network.
With appropriate software, AES-24 commands are capable of being carried by most modern transport networks.

1 Scope

This standard is intended to make it possible to control and monitor, via a digital data network, different audio devices from disparate manufacturers using a unified command set within a standard format.

This standard specifies an extensible application protocol for controlling and monitoring audio devices via digital data networks. AES-24 is the name assigned by this standard to the protocol described herein. The AES-24 protocol provides a relatively complete set of commands and responses for common audio devices, and is designed to be extensible in an orderly fashion to accommodate new commands and responses as required to suit new audio devices that are invented.

As an application protocol, AES-24 is not directly concerned with the mechanics of data transmission. The AES-24 protocol specifies the formats, rules, and meanings of audio control and monitoring commands, but makes only rudimentary assumptions about how these commands are physically transported from one device to another. Thus, AES-24 is able to operate over a variety of modern data communication facilities.

AES-24 does not specify methods for transmitting audio signals in any form.

2 Normative references

Normative references contain provisions that, through reference in this text, constitute provisions of this document. At the time of publication, the editions indicated were 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.

There are no normative references for this standard. References to subsequent parts of the AES24 standard that are now in draft stage are under consideration. These include AES24-2, *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*; AES24-3, *AES standard for sound system control — Application protocol for controlling and monitoring audio devices via digital data networks — Part 3: Transport requirements*; and AES24-4, *AES standard for sound system control — Application protocol for controlling and monitoring audio devices via digital data networks — Part 4: Internet protocol (IP) transport of AES-24.*

3 Definitions

3.1 Networks

3.1.1

digital data network
network (where the context is clear)
set of entities that communicate by digital means, using facilities and methods that allow any entity to exchange information with any other (subject to access controls), and where the number of entities that may participate is reasonably open-ended

3.1.2

network node
addressable hardware location in a digital data network, such as a network card

3.1.3

transport network
data communications facility that can transmit arbitrary data elements (bit strings) from one entity to another with reliable error detection, and under the control of an orderly addressing scheme

3.1.4

transport node
node (where the context is clear)
addressable logical source or destination in a transport network, where there may be multiple transport nodes per network node

3.1.5

transport address
bit string that uniquely identifies a transport node within its transport network

3.1.6

transport protocol
protocol (such as UDP/IP) by which transport nodes communicate transport-level information

3.1.7

broadcast message
packet of data that is sent by one transport node, and directed toward all neighboring transport nodes

3.1.8

internetwork
collection and interconnection of disparate transport networks (that is, subnetworks)

3.1.9

subnetwork
network (where the context is clear)
disparate transport network in the context of an internetwork

3.1.10

end-point subnetwork
subnetwork that is connected to a main or central subnetwork, and that does not provide a communication path between other subnetworks

3.1.11

central subnetwork
the topologically central subnetwork in a centralized internetwork

3.1.12

centralized internetwork
topologically central subnetwork connecting one or more end-point subnetworks

3.1.13

heterogeneous internetwork
internetwork where at least two subnetworks use disparate transport-level protocols

3.1.14

AES-24 transport node
transport node to which one AES-24 device is attached

3.1.15

AES-24 subnetwork
collection of AES-24 transport nodes reachable by a single transport-level broadcast message without forwarding or routing

3.1.16

AES-24 internetwork
centralized heterogeneous internetwork of AES-24 subnetworks

3.1.17

AES-24 internetworking
internetworking (where the context is clear)
configuration and operation of AES-24 internetworks

3.2 Protocols

3.2.1

protocol
set of rules that define the formats, values, and communications procedures for the orderly exchange of information in an internetwork

3.2.2

application protocol
protocol that supports a particular range of application functions

3.2.3

AES-24
application protocol for controlling and monitoring audio systems

3.3 Objects

3.3.1

object
entity that represents data and an associated set of actions that act on the data; an instance of a class

NOTE In AES-24, an object is an abstraction that models network-controllable audio and nonaudio functions of audio and related equipment.

3.3.2

functional object
object that represents an application function, for example, a gain control, a level detector, or a power switch

3.3.3

user interface object
object that represents a user-interface function, for example, a graphic user-interface control element, a graphic user-interface indicator element, a hardware potentiometer, or a pilot lamp

3.3.4

intermediate object
object that represents a processing element that operates on control and monitoring messages passing between other objects

3.3.5

management object
object that represents an AES-24 housekeeping or utility function, as opposed to an audio or audio-related function

3.3.6

device manager
management object that provides the interface to the device’s distributed registry services, that is responsible for AES-24 device initialization, and that mediates the device's relationship with the rest of the AES-24 internetwork

3.3.7

signal flow manager
management object that holds and in some cases changes the configuration of audio signal processing elements and paths within the device

3.3.8

security manager
management object that implements device and object access control

3.3.9

object name
text name assigned to an AES-24 object that distinguishes it from other objects within a particular device

3.3.10

devicename
text name assigned to a device which distinguishes it from other devices within a particular subnetwork

3.3.11

subnetwork name
text name assigned to a subnetwork which distinguishes it from other subnetworks within a particular internetwork

3.3.12

object path
structured text name that unambiguously identifies a particular object within an AES-24 internetwork

3.3.13

device path
structured text name that unambiguously identifies a particular device within an AES-24 internetwork

3.3.14

object handle
16-bit unsigned integer that corresponds to a particular object name and unambiguously identifies a particular object within an AES-24 device

3.3.15

device handle
16-bit unsigned integer that corresponds to a particular devicename and unambiguously identifies a particular device within an AES-24 subnetwork

3.3.16

subnetwork handle
16-bit unsigned integer that corresponds to a particular subnetwork name and unambiguously identifies a particular subnetwork within an AES-24 internetwork

3.3.17

object address
either an object path, or a 48-bit unsigned number that is the concatenation of (from high-order byte to low-order byte) the subnetwork, device, and object handles, and that unambiguously identifies a unique object within an internetwork

3.3.18

device address
either a device path, or a 32-bit unsigned number that is the concatenation of (from high-order byte to low-order byte) the subnetwork and device handles, and that unambiguously identifies a unique device within an internetwork

3.3.19

subnetwork address
either a subnetwork name or a subnetwork handle

3.3.20

AES-24 interface
set of objects that a product makes available to an AES-24 internetwork for control and monitoring purposes

3.4 Devices

AES-24 device
device (where the context is clear)
set of objects with a related set of functions and a specific common set of management objects

3.5 Gateways

3.5.1

AES-24 gateway
specialized AES-24 device that interconnects two AES-24 subnetworks, and provides internetwork message routing and transport-level translation services

3.5.2

gateway manager
management object that provides the interface to a gateway, that is responsible for gateway initialization, and that mediates the gateway's relationship with the rest of the AES-24 internetwork

3.6 Registries

3.6.1

registry
centralized or distributed registry

3.6.2

registry server
optional centralized service in an AES-24 internetwork that is dedicated to an AES-24 subnetwork, and that assigns and confirms device and subnetwork handles, maps device and subnetwork paths to device and subnetwork addresses, and provides transport address resolution services

3.6.3

distributed registry
distributed service in an AES-24 internetwork that maps device and subnetwork paths to device and subnetwork addresses, and provides transport address resolution services

3.6.4

registry server
specialized AES-24 device that implements a centralized AES-24 registry for an AES-24 subnetwork

3.6.5

registry manager
management object that provides the interface to a registry server, that is responsible for registry server initialization, and that mediates the registry server's relationship with the rest of the AES-24 internetwork

3.6.6

register
act of assigning and confirming a device or subnetwork handle with a registry server or, in the case of a distributed registry, with other device and gateway managers

3.6.7

deregister
act of releasing a device or subnetwork handle through a registry server or, in the case of a distributed registry, with other device or gateway managers

3.7 Classes

3.7.1

class
template that defines a particular type of object

3.7.2

instance
term used for object to emphasize the nature of the object as an entity created from its class, as in “object XX is an instance of class YY”

3.7.3

inheritance
relationship between classes in which each class is considered to be a specialized entity derived from another, less specialized one

3.7.4

class hierarchy
set of all AES-24 classes, arranged hierarchically by inheritance

3.7.5

class identifier
unique identifier for an AES-24 class

3.7.6

subclass,
child class
inferior hierarchical relationship between classes

3.7.7

superclass,
parent class
superior hierarchical relationship between classes

NOTE If class X is a specialization of class A, then it is said that X is a subclass of A, and A is a superclass of X.

3.7.8

proprietary class
nonstandard class that a manufacturer adds to the class hierarchy

3.7.9

root class
basic parentless class from which all others are derived

3.7.10

functional class
class representing functional objects

3.7.11

functional matrix class
class representing a matrix of functional objects

3.7.12

user interface class
class representing user-interface objects

3.7.13

user interface matrix class
class representing a matrix of user-interface objects

3.7.14

intermediate class
class representing intermediate objects

3.7.15

network management class
class representing network management objects

3.7.16

actuator class
class representing functional objects that affect signals or power sources in some way, such as switches, filters, attenuators, and so on

3.7.17

sensor class
class representing functional objects that report the value of signals or power sources in some way, such as level detectors, binary indicators, and so on

3.7.18

security manager class
class representing security management objects

3.7.19

signal flow manager class
class representing signal flow objects

3.7.20

binary large object
BLOB
object that sends or receives large formatted blocks of binary data

3.8 Messages and data

3.8.1

message
formatted block of data that is exchanged between objects

3.8.2

property
internetwork-accessible data element belonging to an AES-24 object

3.8.3

property ID
unique identifier of a property within a class

3.8.4

status code,
return code
integer value that indicates the outcome of a previous message

3.9 Methods

3.9.1

method
procedural interface to an object that requires specific parameters, performs a specific action, and may provide specific status information when the function is completed

3.9.2

property access method
PAM
method that operates directly on a property

3.9.3

class-specific method
CSM
method that is unique to its defining class

3.10 Events

3.10.1

event
defined transient state of an object that can cause it to emit one or more AES-24 messages

3.10.2

event table
structured property that defines the message or messages that an object's event or events emit

3.10.3

event message
AES-24 message that is generated by an event

3.10.4

event ID
unique identifier of an event within a class

3.10.5

blocking
act of waiting for a task to complete before continuing with other tasks

3.10.6

non-blocking
act of continuing to execute other tasks while expecting results from a previously initiated task

3.10.7

closed event processing sequence
blocking event processing sequence in which the event message or messages are sent with reply requested

3.10.8

open event processing sequence
non-blocking event processing sequence in which the event message or messages are sent with no reply requested

3.11 System initialization and configuration

3.11.1

subnetwork device inventory
inventory (where the context is clear)
set of devices within a subnetwork

3.11.2

internetwork device inventory
union of all the subnetwork inventories in an internetwork

3.11.3

object discovery
act of enumerating all objects within a device

3.11.4

subnetwork device discovery
act of enumerating all devices within a subnetwork

3.11.5

internetwork device discovery
act of enumerating all devices within an internetwork

3.11.6

AES-24 device initialization
process by which a device joining an AES-24 subnetwork configures itself so that it can interoperate with other devices in the internetwork

3.11.7

AES-24 registry initialization
process by which a registry (a specialized device) joining an AES-24 subnetwork configures itself so that it can interoperate with other AES-24 devices in the internetwork

3.11.8

AES-24 gateway initialization
process by which a gateway (a specialized device) joining an AES-24 internetwork configures itself so that it can interoperate with other devices in the internetwork

3.11.9

system builder
software tool that initializes event tables and other operating parameters of an AES-24 internetwork

3.12 System controllers

3.12.1

system controller
component whose primary role is to control and monitor the audio processing functions on an AES-24 network

3.12.2

generic controller
system controller that contains a system builder capable of automatically or semi-automatically generating controller configurations and sets of control paths for a relatively wide range of equipment types

4 General requirements

4.1 Function

The function of equipment conforming to this standard shall be to make possible the control and monitoring, via a digital data network, of different audio devices from disparate manufacturers using a unified command set within a standard format. The AES-24 protocol provides a relatively complete set of commands and responses for common audio devices, extensible in an orderly fashion to accommodate new formats as required to suit new audio devices as they are invented.

4.2 Messages

Under AES-24, control and monitoring shall be carried out by messages that pass back and forth between objects. AES-24 internetwork traffic shall consist entirely of messages between objects.

4.3 Classes

4.3.1 The design of each kind of object shall be specified by a formalized abstraction named a class. Each class shall be the template for a particular kind of object, and each object shall be an instance of its defining class. Each class can have many instances, but each object shall be an instance of one and only one class.

4.3.2 The class of an object shall define the messages it can send and receive, and the meanings and effects of those messages.

4.3.3 The set of all standard classes shall be organized into a class hierarchy in which special-purpose classes are defined using general-purpose classes as a basis.

4.3.4 Each class shall have its own unique identifier.

4.3.5 Each class shall define the following elements:

a) set of properties (see 5.1);
b) set of methods (see 5.2);
c) set of events (see 5.3).

NOTE A high-level overview of the AES-24 class hierarchy is presented in 8.3. Class details are considered in AES24-2.

5 Objects

Every AES-24 object shall be an instance of a particular class. The object's class completely defines the object's methods, properties, and events. Every object of a given class shall have exactly the same set of methods, the same set of properties, and the same set of events. Therefore, knowing an object's class is sufficient for knowing its entire control and monitoring interface. In addition, each object (that is, each instance of a class) shall include

a) a name and a code that each uniquely identifies the object within the device;
b) the particular data that define the object's state.

5.1 Properties

AES-24 objects may contain data elements known as properties whose values are accessed via PAMs.

The purpose of a property shall be to present an operational parameter of the object to the AES-24 internetwork so that other objects may retrieve it and, except for read-only parameters, change it. In a normal AES-24 object, the set of properties shall be as complete as possible, that is, the manipulation of properties through PAMs shall encompass most, if not all, of the object's defined function. In normal AES-24 internetworks, the bulk of the message traffic should be concerned with the invocation of PAMs and, hence, the manipulation of property values.

NOTE Details on the format and content of these property access methods, and their IDs, are considered in AES24-2.

5.2 Methods

Every object shall present a set of functional interfaces to the internetwork. These interfaces, called methods, provide a procedural interface that other objects may invoke via specific AES-24 messages.

A method shall be completely defined by:

a) the parameters it accepts;
b) the actions it takes;
c) the responding messages it may provide.

Every AES-24 message shall be either an invocation of a particular method or a method's response to a previous message.

NOTE The specific coding schemes used in AES-24 messages to identify methods, responses, and so on are described in clause 13.

Methods are broadly divided into two categories, property access methods and class-specific methods. PAMs are used to read, write, and manipulate object property values. CSMs are used to carry out operations specific to their object’s defining class.

5.3 Events

Certain transient states of an object can cause it to emit one or more AES-24 messages. These states are called "events". See 3.10.1.

An event is said to occur in or be raised by a particular object when that object enters the event state.

An object's class shall define the object's repertoire of events, that is, which specific states shall cause messages to be emitted.

The message or messages that an object's event or events can emit shall be defined by a structured property known as the event table. Such messages are called event messages.

Event messages shall be standard AES-24 messages, indistinguishable from messages from sources other than events. The event table shall control the format, contents, and destination of each event message. An event message may be successfully directed to any method in any object, as long as the event table entry involved has been set up to cause the event message to be generated in the form required by the destination method.

Except where noted for specific classes, the initial contents of an event table shall be null, which implies that any event of a freshly initialized object cannot emit messages. During internetwork initialization, configuration processes should initialize event table entries in the various objects to define all the control relationships that the application requires.

NOTE In general, AES-24 is based on an event-driven programming paradigm, in which events and the messages they emit are the fundamental animating mechanism of the control system. The AES-24 design assumes that most AES-24 internetwork activity should arise directly or indirectly from the occurrence of events and the consequent exchange of event messages. Again, it is expected that most events will be concerned with invoking PAMs.

5.4 Property, method, and event IDs

Every property, method, and event shall be identified by unique identifiers known as the property, method, and event IDs, respectively.

The format of the property, method, and event IDs shall be as shown in Table 1.

Table 1 — Property, method, and event IDs

Field Name	Bits	Meaning
`class_level`	13–10	Level in the AES-24 class hierarchy where the addressed property, method, or event is defined: 0 = Root, 1 = next level down from root
property, method or event identifier	9–0	Unique index to the property, method, or event within the class.

For each class, there is one and only one path of inheritance to the root class. This path is called a *branch* of the class hierarchy. A class shall inherit all the attributes of its ancestors, all of which are in its branch.

The purpose of defining property, method, and event IDs in this manner is to ensure adequate address space for use by new classes to be defined in the future. Note that a property, method, or event ID does not uniquely identify a specific class, since there are multiple classes at the same level, each belonging to a different branch. Therefore, at run time AES-24 applications shall use other means (usually object handles) to identify the specific classes to which messages are addressed.

6 Devices

6.1 Standard devices

In AES-24, a device may be implemented as a single physical unit, although AES-24 does not require this. Conversely, a single physical unit may contain multiple devices. Multiple devices may also share a physical

network node; however, each shall have its own transport node. In other words, each AES-24 device shall be uniquely addressable by the transport network software (such as through port numbers in UDP/IP).

The contents of a device shall include:

a) zero or more functional or functional matrix objects. This set includes objects to represent all of the device's application functions that are subject to AES-24 control and monitoring;
b) zero or more user interface or user interface matrix objects. This set includes objects to represent all of the device's user interface functions that are used for AES-24 control and monitoring;
c) zero or more intermediate objects. This set includes objects to represent all of the device's processing services that operate on control and monitoring commands flowing between functional and user interface objects;
d) zero or more binary large objects. This set includes objects used to modify device status (such as through presets), or to upload binary program information (such as for a programmable DSP device);
e) an optional signal flow manager. The signal flow manager's functions are to be developed;
f) exactly one device manager;
g) an optional security manager.

The device manager shall manage the addressing of objects, performing all initialization activities needed to establish intercommunication between the objects within the device and the other objects of the internetwork. In dynamically configurable devices, device managers shall have the power to create other objects, including other device managers. The security manager's functions are under consideration.

The device model is illustrated in Figure 1.

NOTE The demographics of the object populations of devices will vary depending on the intended purposes of the devices. Generally speaking, devices whose purposes are audio processing may have many functional objects, yet few (or no) user interfaces or intermediate objects. Devices whose purposes are system control may have many user interfaces and intermediate objects, but few (or no) functional objects.

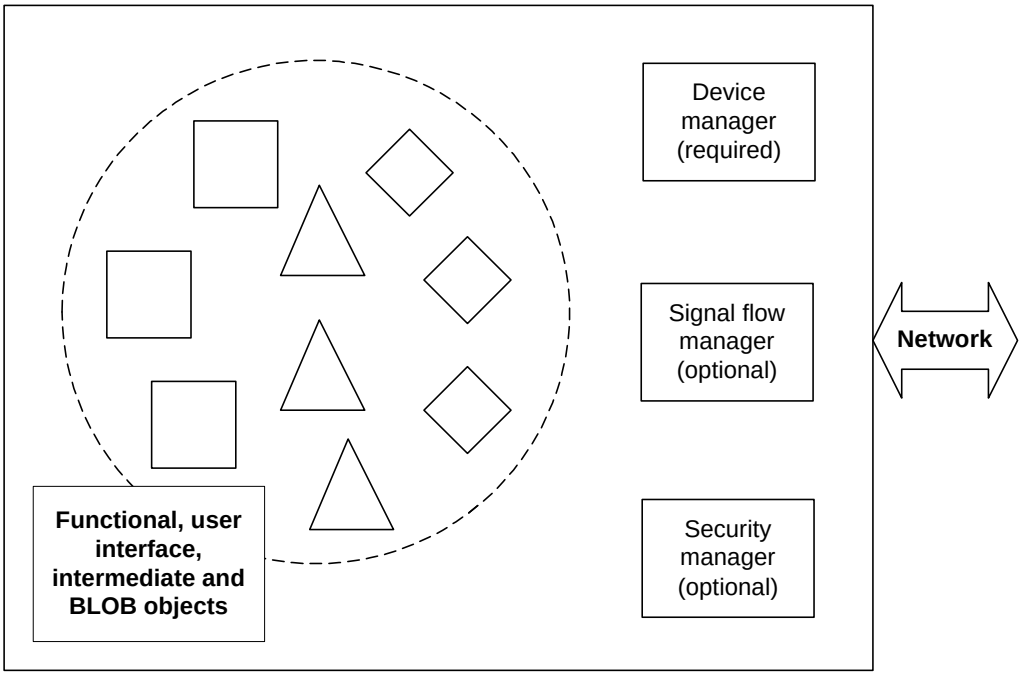

Figure 1 — AES-24 device model

6.2 Registry servers

An AES-24 registry server shall be dedicated to the subnetwork in which it resides, that shall assign and confirm device and subnetwork handles, shall map device and subnetwork paths to device and subnetwork addresses, and shall provide transport address resolution services. Instead of a device manager object, a registry server shall have a registry manager object. See clause 10 for details on this device.

6.3 Gateway devices

An AES-24 gateway shall be responsible for directing AES-24 message traffic between the central subnetwork and an end-point subnetwork in an AES-24 internetwork. Instead of a device manager object, a gateway device shall have a gateway manager object. See clause 7 for details on internetworking details.

7 Networking

7.1 Subnetworks

An AES-24 subnetwork shall consist of the following:

a) a set of AES-24 devices;
b) an optional registry server to assign and confirm device and subnetwork handles, map device and subnetwork paths to device and subnetwork addresses, and provide transport address resolution services;
c) a transport network (cabling and transport-level networking protocol software).

AES-24 subnetworks shall be peer-to-peer. This requirement means that any device can control and monitor any other. Figure 2 shows an AES-24 subnetwork.

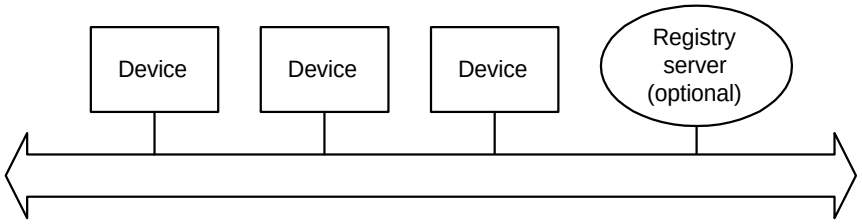

Figure 2 — AES-24 subnetwork

Although Figure 2 represents a simple network topology, AES-24 supports the more sophisticated configurations that can be realized with the use of network bridges, hubs, and repeaters, provided that each device shall be reachable by a transport-level broadcast message generated by any other device on that subnetwork.

7.2 Internetworks

AES-24 subnetworks can be interconnected to form an AES-24 internetwork, as shown in Figure 3. The primary reason for supporting internetworking is to allow AES-24 subnetworks using disparate transport networks to intercommunicate. It can also be useful when geographical restrictions require the use of internetwork routers to realize a connection.

AES-24 supports centralized heterogeneous internetworks. That is, an AES-24 internetwork's topology shall be restricted to a centralized configuration. Specifically, one subnetwork shall be identified as the central subnetwork, while one or more other subnetworks (termed end-point subnetworks) shall be connected to the central subnetwork through an AES-24 gateway as shown in Figure 3.

NOTE Details on AES-24 gateways are considered in AES24-2.

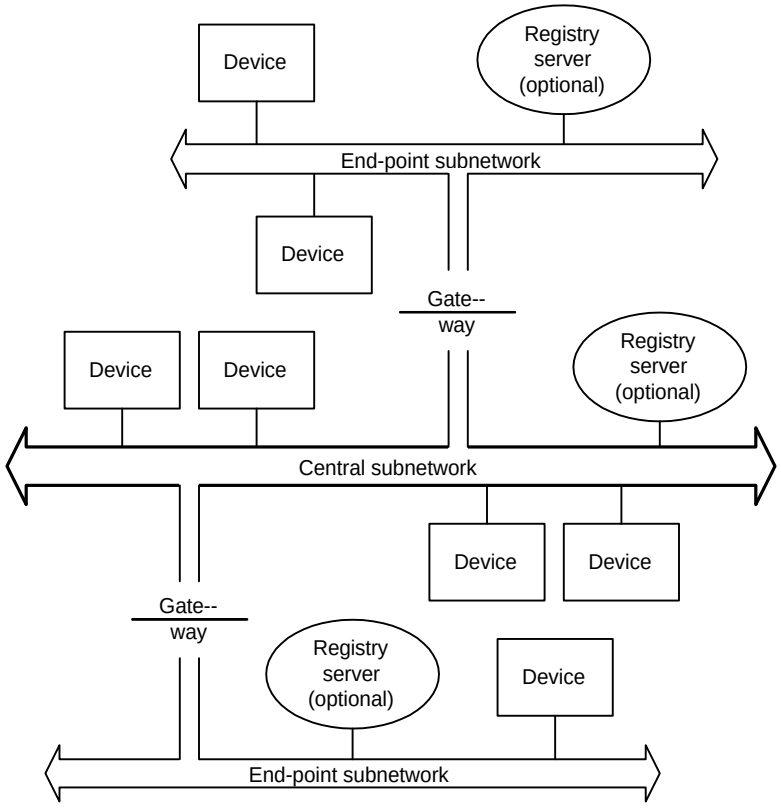

Figure 3 — AES-24 internetwork

NOTE Because an AES-24 gateway, like a registry server, shall keep AES-24-to-transport address mapping tables for message routing purposes, it is possible, although not necessary, that it can provide a centralized registry service for its end-point subnetwork. This type of registry server may be referred to as an end-point registry server. Similarly, the central subnetwork's registry server may be referred to as the central registry server.

8 Classes

AES-24 classes shall be arranged in a hierarchy in which the lower classes are increasingly specialized versions of the higher ones. This derivative relationship is termed inheritance.

The root class at the highest level of the class hierarchy shall be the single exception, in that it is not derived from a less specialized class.

The less specialized class to which a class is related is called its parent, parent class, or superclass; the more specialized class or classes to which a class is related are called children, child classes, or subclasses.

8.1 Elements

8.1.1 Root class

The root class shall have the following elements:

a) a set of methods;
b) a unique class identifier.

8.1.2 All other classes

Except for the root class, all AES-24 classes shall have the following elements:

a) a set of methods;
b) a unique class identifier;
c) exactly one parent class.

8.1.3 Inheritance

In the AES-24 class hierarchy, inheritance shall be defined according to the following rules:

a) the method set of a child class shall include all the methods of its parent class;
b) a child class may expand the definitions of one or more of its parent class's methods;
c) a child class may define methods of its own;
d) any given class except the root class shall inherit from exactly one other class.

8.2 Maintenance

The AES-24 class hierarchy shall be a detailed and specific reflection of common audio equipment functions now and in the future. The means by which the class hierarchy shall be updated is considered in AES24-2.

This maintenance process shall attempt to balance completeness of representation against clarity and elegance of the class hierarchy.

8.2.1 Maintenance rules

New classes shall be defined on a case-by-case basis in accordance with the following rules:

a) no class shall ever be deleted from the class hierarchy;
b) every new class, when added, shall be added as a child class only, that is, no classes will be interposed between existing parents and children;
c) if a new application function can reasonably be supported by employing one or more objects from existing classes, no new class shall be defined for it; however, objects that are fundamentally different in application function from those defined by existing classes, or that require new messages, generally justify the creation of new classes;
d) whenever a new class is defined, it shall be created as a child of the most specialized class that reasonably supports the basic nature of the new function or functions;
e) a new class shall not redefine the meaning of any inherited method;
f) AES-24 implementations shall use the most specific classes that reasonably express their functions.

These maintenance rules are intended to ensure that class hierarchy updates shall continue to support the three types of compatibility (upward, downward, and lateral) articulated as AES-24 design goals. These rules shall be followed not only by those maintaining the official standard but also by manufacturers adding nonstandard classes to the class hierarchy who wish to maximize interoperability with standard products.

8.2.2 Nonstandard classes

From time to time, manufacturers can find it necessary to extend the standard class hierarchy to support new or special products or functions that AES24 does not currently address. Such extensions shall be made in the form of proprietary classes added to the class hierarchy. Proprietary classes do not form a part of the standard AES-24 protocol, but will generally interoperate with it if they are defined in accordance with 8.2.1.

Manufacturers shall define proprietary classes to support product functions or performance requirements not supported by the standard classes. When a particular proprietary class moves into general use, its owner may, according to means considered in AES24-2, request standardization of the proprietary class by adding it to the standard class hierarchy.

8.3 Class hierarchy

The AES-24 class hierarchy is summarized in Figure 4. Each box represents a class. The connecting lines represent inheritance relationships, which in this picture proceed downward. The complete description of the AES-24 class hierarchy and all its classes is considered in AES24-2.

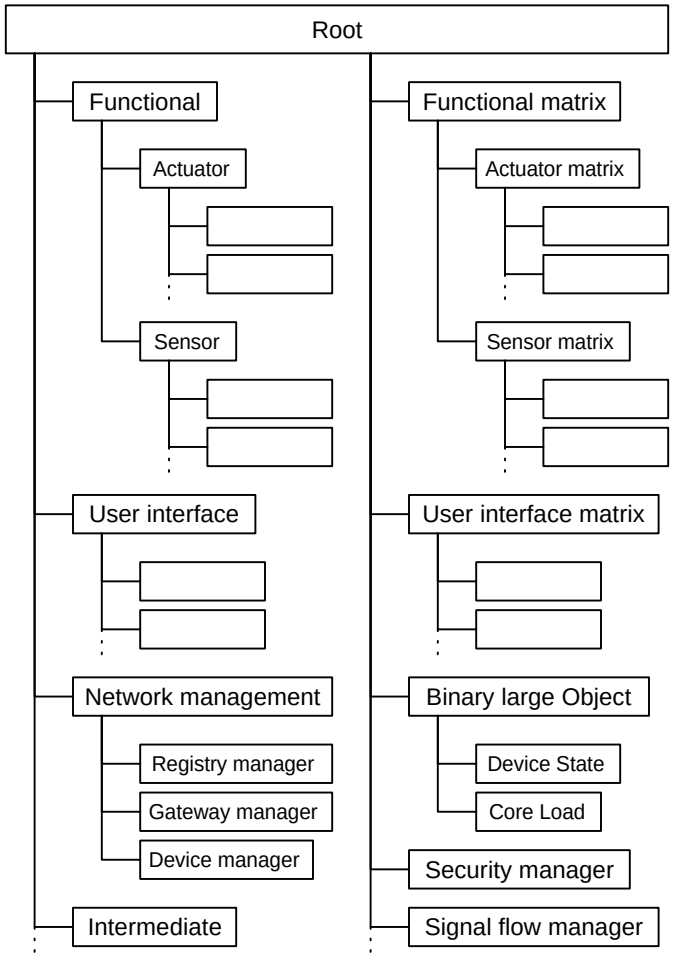

Figure 4 — AES-24 class hierarchy (summary)

Figure 4 shows only the topmost levels of the class hierarchy. Levels below the top are indicated schematically by the unlabeled boxes, which are intended to suggest the various specific types of actuators, sensors, controls, and indicators that AES-24 defines. In the complete class hierarchy considered in AES24-2, there may be several hierarchical levels of classes below the ones shown. The classes shown in Figure 4 are

a) root, the basic parentless class from which all others are derived;
b) functional, functional matrix, user interface, user interface matrix, intermediate, and management; the general classes representing the major categories of AES-24 objects;
c) actuator, a class representing functional objects that affect signals or power sources in some way — switches, filters, attenuators, and so on;
d) sensor, a class representing functional objects that report the values of signals or power sources in some way — level detectors, binary indicators, and so on;
e) registry, described in clause 10;
f) device manager, signal flow manager, and security manager; classes representing the standard management objects.

9 Addressing

AES-24 shall support only object-to-object communication. Thus, every object shall have its own address. AES-24 object addresses shall have two forms:

a) an object path, which shall be a preassigned text string;
b) an object address, which shall be a 48-bit unsigned integer.

Both an object path and an object address unambiguously represent a specific object in a specific device, in a specific subnetwork of an AES-24 internetwork.

9.1 Identifiers

Each identifier shall:

a) be case insensitive;
b) begin with an underscore (_) or letter (where letters shall be language specific);
c) contain only underscores, letters, and digits (0...9);
d) not contain more than 254 underscores, letters, digits, or combinations thereof.

Each object shall have an identifier, known as the object name, that shall be unique within the device in which it resides.

Each device shall have an identifier, known as the devicename, that shall be unique within the subnetwork in which it resides.

Each subnetwork shall have an identifier, known as the subnetwork name, that is unique within the internetwork in which it resides.

9.2 Paths

9.2.1 Object paths

An object path shall be formed by combining a subnetwork name identifier, devicename identifier, and object name identifier as follows:

```
[//SUBNETWORKNAME/]DEVICENAME/OBJECTNAME
```

NOTE 1 The brackets around `//SUBNETWORKNAME/` are used to denote that the subnetwork name is optional when referencing an object that is in the same subnetwork. If used, its value shall be set during gateway initialization. See 11.3.

NOTE 2 If the context is clear, leading slashes may be omitted. Details are considered in AES24-2.

NOTE 3 Forward and backward slashes may be used interchangeably; therefore, `/\NET1\DEV1/OBJ1` is a legal object path.

9.2.2 Device paths

A device path shall be formed by combining a subnetwork name identifier and a devicename identifier as follows:

```
[//SUBNETWORKNAME/]DEVICENAME
```

9.3 Object, device, and subnetwork name assignments

9.3.1 Object name

Each object name shall be set at the time the device is manufactured or, for devices whose function is user-definable (for example, programmable digital audio processors), at the time the object is created. For a device manager, the object name shall be the reserved word `DEVICE`. Similarly, a gateway manager shall be named `GATEWAY`, and a registry manager shall be named `REGISTRY`. Object names, other than `DEVICE`, `GATEWAY`, and `REGISTRY`, may be reassignable at runtime. Reassigned object names may be volatile. Details are considered in AES24-2.

9.3.2 Devicename

In an initialized AES-24 network, every AES-24 device shall have a unique devicename. The initialization process shall ensure that each devicename within the network is unique by causing new devicenames to be assigned where necessary.

Depending on the implementation of a device, the initial value of its devicename may be writable or non-writable. Every device, however, shall have its initial devicename uniquely assigned at time of manufacture. The device's manufacturer shall allocate such devicenames in cooperation with the AES.

If a devicename clash occurs during initialization or a rename request is received, the subject device shall adopt a new devicename. If the device has a writable devicename, the new devicename may be saved in non-volatile storage, causing the device to assert the same newly defined name on subsequent initializations. If the device has a nonwritable devicename, the new devicename will survive only until reset or power-off, at which time the device's devicename will revert to its permanent, manufacturer-assigned value.

The matter of swapping devices in a running network is addressed in annex C.

9.3.3 Subnetwork name

Subnetwork names shall be provided by AES-24 gateway devices. They shall be assigned uniquely at the time of manufacture, and shall be both rewritable and non-volatile. Assignment details are provided in 11.3 and considered in AES24-2.

9.4 Handles

9.4.1 AES-24 address format

There are three types of handles, namely, subnetwork, device, and object handles. They shall correspond directly with subnetwork names, devicenames, and object names. When used collectively, they shall uniquely identify an object internetwork wide. Their format is as shown in Figure 5.

16 bit	16 bit	16 bit
Subnetwork Handle	Device Handle	Object Handle

Figure 5 — AES-24 address format

The subnetwork handle shall identify the subnetwork on the internetwork where a device or object lives. The device handle shall identify the device on a specific subnetwork within which an object lives. Finally, the object handle shall identify a specific object within a specific device. Subnetwork and device handles shall be allocated during AES-24 initialization or configuration.

A registry server shall not participate in device handle arbitration since it shall have a default address of 0000_{16}. For that reason, the registry server shall not have a device manager, rather it shall have a registry manager.

9.4.2 Default handle values

Default handle values shall be as provided in Table 2.

Table 2 — Default handles

Subnetwork handles		Device handles		Object handles	
Local subnetwork	0000_{16}	Active registry server	0000_{16}	Registry, device, or gateway manager	0000_{16}
Central subnetwork	0001_{16}	Gateway	0001_{16}	All objects	$FFFF_{16}$
All subnetworks	$FFFF_{16}$	All devices	$FFFF_{16}$	Reserved	$FFFE_{16}$–$FFF0_{16}$
Reserved	$FFFE_{16}$–$FFF0_{16}$	All registry servers	$FFFE_{16}$		
		All gateways	$FFFD_{16}$		
		Reserved	$FFFC_{16}$–$FFF0_{16}$		

For clarity, note the following example addresses and their references (this is not a complete list):

SSSSDDDDOOOO	S = subnet handle; D = device handle; O = object handle
000000000000	Active registry server on local subnetwork
000100000000	Active registry server on central subnetwork
0000FFFF0000	All device managers on local subnetwork
0000FFFD0000	All gateway managers on local subnetwork (that is, all internetwork gateways if referenced from central subnetwork, or the end-point gateway if referenced from an end-point subnetwork)
0001FFFD0000	All gateway managers (since all gateways shall be attached to the central subnetwork)
000100010000	Invalid (no gateway shall be dedicated to the central subnetwork)
000000010000	Gateway manager on local subnetwork (shall be invalid if referenced from central subnetwork)
FFFFFFFFFFFF	Every object in the entire internetwork

An end-point subnetwork and device handle shall be valid until its respective gateway or device either powers down or releases it for reuse, whichever comes first.

10 Registries

10.1 Registry

Every AES-24 network shall have a registry. The registry service may be provided either by a single centralized server or in a distributed fashion by the device managers.

All registries, whether distributed or centralized, shall provide the following services:

a) managing and assigning device and subnetwork handle values;

b) providing subnetwork and (optionally) internetwork device inventory information.

The registry shall keep records of device and subnetwork handles as well as the device and subnetwork paths they respectively represent. The registry shall allocate and deallocate (register and deregister) device and subnetwork handles on request.

The registry provides a repository for network configuration information. Objects that need this information can query the registry to retrieve it. The process of retrieving a network's inventory is called discovery.

10.2 Registry server

In addition to the items in 10.1, a registry server shall provide the following two services:

a) notifying the subnetwork and (optionally) the internetwork of its subnetwork device inventory changes;
b) polling the subnetwork periodically to ensure that registered device managers and gateways are still functioning.

When a subnetwork is initialized, or when a running subnetwork's configuration changes, the registry shall broadcast a notification message to its subscribers. Any device in the network may be a subscriber by sending an appropriate request to the registry.

The frequency of the polling shall be at the discretion of the implementor.

10.3 Stand-by registry server

Stand-by registry servers are essentially used as backup registry servers in case the active registry server fails. It is the responsibility of a stand-by registry server to periodically poll the active registry server to confirm its presence. If after a sufficient time period (defined by the registry server manufacturer), there is no response from the active registry server, a stand-by registry server shall attempt to become the active registry server. Because there may be multiple stand-by registry servers, such attempts shall be made in cooperation with the other stand-by registry servers. See 11.2.

10.4 Distributed registry

A distributed registry is realized through standard functions that shall be required of standard AES-24 devices and AES-24 gateways. Details are considered in AES24-2.

10.5 Registry scope

The scope of every registry shall be exactly one subnetwork, and each subnetwork shall have exactly one registry at any given time. Regardless of whether it is distributed or centralized, the registry shall manage devicenames and device handles. Additionally, the registry on the central subnetwork shall manage subnetwork names and subnetwork handles.

The registry services shall ensure the uniqueness of each registered name and handle within the registry's subnetwork and, in the case of subnetwork names and handles, their uniqueness throughout the centralized internetwork.

Every registry shall enforce a strict one-to-one correspondence between name and handle values.

10.6 Using the registry

Subject to access controls, which are under consideration, any object in the network may initiate registry transactions.

Objects that use the registry and their functions shall be as follows.

10.6.1 Device managers

When any device powers up or resets, that device's device manager object shall be responsible for registering itself in the registry.

10.6.2 System builders

These are objects that configure AES-24 networks. See 11.3

10.7 Distributed versus centralized registry

In a subnetwork with a centralized registry (one with a registry server), device and subnetwork handle management shall be accomplished by means of query and update transactions with its active registry server. In a subnetwork with a distributed registry (one without a registry server), device and subnetwork handle management shall be accomplished, respectively, by subnetwork-wide and internetwork-wide negotiation.

NOTE Multiple broadcast messages for this purpose are considered in AES24-2.

It is expected that a centralized registry will be more reliable and much more efficient than a distributed registry.

Registry servers may be indicated for networks with some or all of the following attributes:

a) many devices;
b) frequent and dynamic device reconfiguration;
c) low bandwidth transport;
d) stringent initialization time requirements.

For a device to be AES-24 compliant, it shall operate with both distributed and centralized registries. It is understood that in networks with distributed registries, the device can perform some functions more slowly.

A device shall know that a registry server exists, and therefore shall use its services, under either of the following conditions:

a) a registry server assigns or confirms its device handle during device initialization;
b) a registry server makes its presence known through its standard device polling sequence.

A device shall know that a registry server does not exist, and therefore shall use distributed registry services until the registry server reestablishes its presence, under any of the following conditions (see 11.1):

a) a registry server does not respond during device initialization;
b) a registry server posts an orderly shutdown message;
c) a registry server does not respond within *x* seconds (a time period that shall be defined by the device manufacturer) after the device attempts to use its services.

10.8 Registry reliability

Registries are not involved in exchange of operational control and monitoring messages in AES-24 networks. Once a subnetwork is initialized, and as long as no dynamic configuration changes are pending, its registry server shall resort to the periodic polling of registered devices to assure their presence. Therefore, registry server failure cannot impact network function until the network is reinitialized or reconfigured.

To reduce single point of failure exposures of subnetworks with registry servers, AES-24 shall support the existence of stand-by registry servers. These shall be registry servers that sit passively on a subnetwork, awaiting failure of the active registry server. A subnetwork may have any number of stand-by registry servers.

If a registry server fails and there is no stand-by registry server, all devices on its subnetwork shall switch to distributed registry mode as indicated in the 10.7. If this switching occurs, reconfiguration performance may degrade, but normal operations shall not be affected.

11 AES-24 initialization

When a device (a new device in what follows) is first connected to an AES-24 network, certain actions shall occur so that the device can successfully interoperate with the other devices in the network. These actions are termed AES-24 initialization.

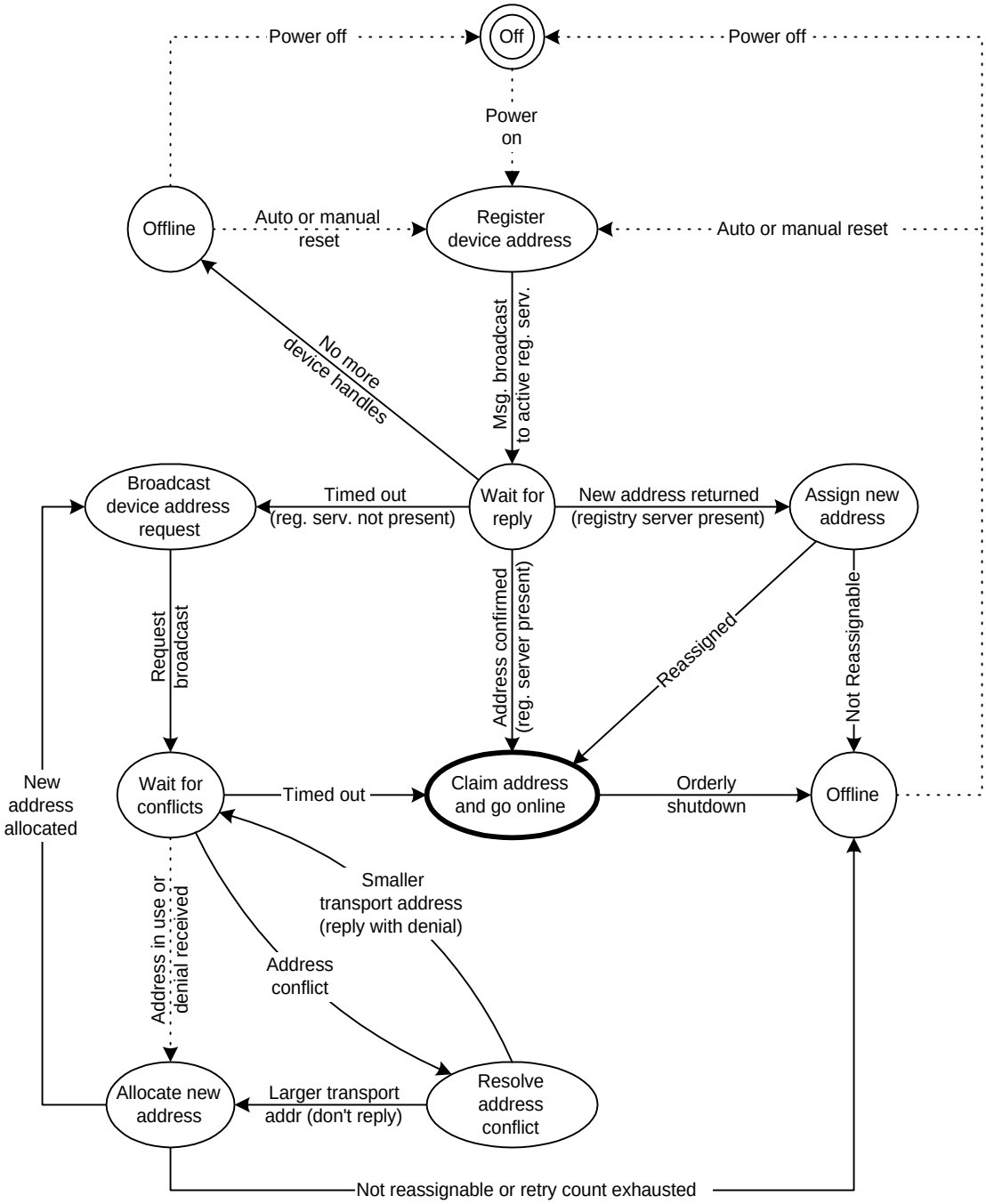

Figure 6 — AES-24 device initialization.

AES-24 initialization is not the same as power-on initialization of a piece of equipment, which is an internal device process not directly related to AES-24.

AES-24 initialization shall have the following main actions:

a) registry identification: the new device manager (that is, the device manager of the device joining the network) shall discover whether there is a centralized or distributed registry;
b) devicename negotiation: the new device manager shall negotiate a unique devicename (or detect an irresolvable clash and abort its initialization);
c) device handle negotiation: the new device manager shall allocate and arbitrate a unique, subnetwork-wide device handle;
d) discovery: if needed, objects in the new device shall learn the handles of other devices' objects;
e) inventory change announcement (performed only if there is a registry server): device managers on the network shall be informed of the new device's entry.

Within each device, the object responsible for AES-24 initialization shall be the device manager.

NOTE Clause 11 describes the most common message exchanges that occur during initialization. In these descriptions, unusual message sequences have been omitted for simplicity. Complete state definition that covers all of the possible initialization sequences, common and uncommon, are shown in Figures 6 – 8.

11.1 Device initialization

During initialization, a device undergoes the processes of registry identification, devicename negotiation, and handle range allocation. All are intertwined.

Figure 6 shows the general initialization sequence of an AES-24 device. Details on the specific message format and return values are considered in AES24-2. Note that in Figure 7, “device address” refers to the device’s name and 16-bit handle.

11.1.1 Device initialization sequence

A device shall begin with a power-on and hardware (or device-specific) initialization sequence that shall proceed as follows:

NOTE All references in this clause to x_i seconds refer to a time period that shall be defined by the device manufacturer.

A device shall broadcast (at the transport level) a devicename and handle registration message to the registry, and shall wait for x seconds for a response. There are two possible outcomes to this.

1) There is a response from the registry. In this case, there are three possible outcomes:

a) The registry responds with a return code indicating that there are no more available addresses (that is, devicenames or device handles). In this case, the device shall go offline.
b) The registry responds with a return code confirming the requested address (both devicename and handle). In this case, the device shall claim the address, and proceed to the online state to begin participating on the network for normal AES-24 control transactions.
c) The registry responds with a return code indicating an address clash (that is, that the address is already in use), and suggests either a new devicename or a new handle or both. The device shall either accept or reject the new devicename or handle. If the device accepts the new address, it shall claim the address, and proceed to the online state. If the device rejects the new address, it shall go offline.

2) There is no response from the registry within x_1 seconds. In this case the device shall assume a distributed registry and shall proceed by broadcasting its name and handle to all device managers on the network. The device shall give sufficient time, x_2 seconds, for other devices to respond. There are three possible outcomes to this:

a) Another device responds with a return code indicating that the address has been denied. An address shall be denied for only one of two reasons:

1) the responding device is already using the desired address (that is, an address clash);
2) the responding device desires the same AES-24 address, and has a numerically smaller transport address.

In either case, the initializing device shall either allocate and attempt to register a new address and proceed as in step 2, or go offline. The device may give up (that is, proceed to the offline state) after a certain number of retries (a number that shall be defined by the manufacturer).

b) Another initializing device makes a request for the same address. A device shall know this by receiving other devices' address request event messages while waiting during the x_2 time period. In this case, the device shall compare its own transport address with that of the other device that is requesting the same address to resolve the conflict. There are two possible outcomes to this:

1) its own transport address is numerically greater than that of the other device, in which case, the device (not the other device) shall either allocate and attempt to register a new address (and proceed as in step 2), or go offline;
2) its own transport address is numerically smaller than that of the other device, in which case, a reply indicating an address deny shall be sent to the other device, and the device (not the other device) shall continue to process more conflicts until the x_2 time period has expired. The other device shall follow as in step 2a2.

c) The x_2 time period expires since neither case 2a nor case 2b1 has been fulfilled. In this case, the device shall assume that the requested devicename and device handle are acceptable and, hence, shall claim the address and proceed to the online state.

Details on device initialization messages and required events are considered in AES24-2.

11.1.2 Non-initialized device communication

During initialization, and before a device address has been confirmed, devices are not assigned a unique AES-24 address; however they shall be guaranteed a unique transport address. Therefore, all responses to device initialization broadcast messages shall be made to the initializing device's unique transport address. This address shall be obtainable from the transport layers (for example, `recvfrom()` in the WinSock API).

NOTE Broadcast messages on IP networks should be delivered to a well-known AES-24 port number. This port number is considered in AES24-4.

11.1.3 Initialized device communication

Initialized devices (that is, those in the online state) shall communicate with other devices in a point-to-point manner whenever possible, that is, from one specific transport address to another. This manner is to reduce network traffic in the upper transport layers. Broadcasting at the transport level to a unique AES-24 device handle is recommended only when the destination-device transport address is not available.

11.2 Registry server initialization

The control-flow diagram in Figure 7 shows how a registry server shall start up and proceed to the active state whereupon it shall service its subnetwork for device registration messages, as well as other network management services. Each subnetwork may have one or more registry servers, however, only one registry

server shall be active on each subnetwork at any given time. The registry server shall have a default handle of 0.

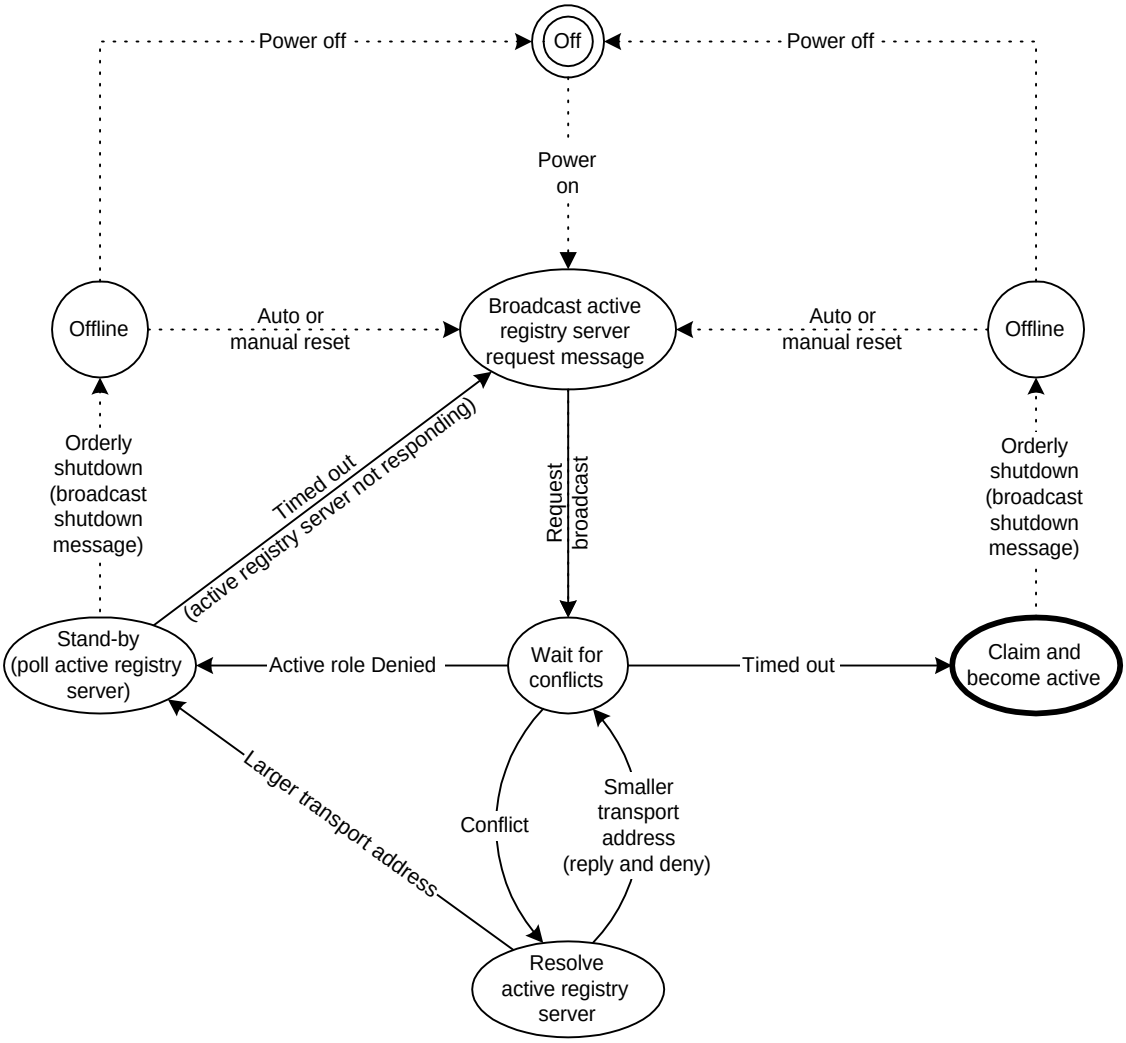

Figure 7 — AES-24 registry initialization

A registry server shall begin with a power-on and hardware (or device-specific) initialization sequence that shall proceed as follows:

NOTE All references to x_i seconds refer to a time period that shall be defined by the device manufacturer.

A registry server shall broadcast (at the transport level) an active-registry-server request message to *all* registry servers (active or stand-by). A registry server shall give sufficient time, x_1 seconds, where x_1 is at least twice that of stand-by registry server polling rate (defined below as x_2), for other registry servers to respond. There are two possible outcomes to this:

1) Another registry server responds with a return code indicating that the request has been denied; this shall occur for only one of two reasons:

a) the responding registry server is already active;
b) the responding registry server also desires to become active, and has a numerically smaller transport address.

In either case, the activating registry server shall proceed to stand-by mode. While on stand-by, a registry server shall periodically check for the presence of the active registry server. If there is no response within x_2 seconds, where x_2 shall be sufficiently long for the active registry server to respond while under heavy load, the stand-by registry server shall restart its registry server arbitration sequence as it does when it is powered on.

2) Another activating registry server makes a request to become active. A registry server shall know this action by receiving other registry servers' active-registry-server-request event messages while waiting during the x_1 time period. In this case, the registry server shall compare its own transport address with that of the other activating registry server to resolve the conflict. There are two possible outcomes to this:

a) Its own transport address is numerically greater than that of the other registry server. In this case, the registry server (not the other registry server) shall proceed to stand-by mode, and shall not restart polling until the active-registry-server-claim message is heard. If it is not heard within x_3 seconds, where x_3 is at least as long as x_2, the stand-by registry server shall immediately restart its registry arbitration sequence as when powered on.
b) Its own transport address is numerically smaller than that of the other activating registry server. In this case, a reply shall be sent to the other activating registry server indicating that its request has been denied. The activating registry server shall continue to process more conflicts until the x_2 time period has expired, while the other activating registry server shall proceed to stand-by mode as described in case 2a.

3) The x_1 time period expires since neither case 1 nor case 2a has been fulfilled. In this case, the activating registry server shall claim the active registry server role and immediately proceed to the active state.

Details on registry server initialization messages are considered in AES24-2.

11.3 Gateway initialization

As shown in Figure 8, gateway initialization shall proceed much like device initialization.

A gateway shall begin with a power-on and hardware (or gateway-specific) initialization sequence that shall proceed as follows:

NOTE All references in this clause to x_i seconds refer to a time period that shall be defined by the gateway manufacturer.

A gateway shall broadcast, onto the central subnetwork, a subnetwork name and handle registration message to the registry and shall wait for x_1 seconds for a reply. There are two possible outcomes:

1) There is a response from the registry. In this case, there are three possible outcomes:

a) The registry responds with a return code indicating that there are no more available addresses (that is, subnetwork names or subnetwork handles). In this case, the gateway shall go offline.
b) The registry responds with a return code confirming the requested address (both subnetwork name and handle). In this case, the gateway shall claim the address and proceed to the online state to begin participating on the network for normal AES-24 control transactions.
c) The registry responds with a return code indicating an address clash (that is, that the address is already in use), and suggests either a new subnetwork name or a new handle or both. The gateway shall either accept or reject the new subnetwork name or handle. If the gateway accepts the new address, it shall claim the address and proceed to the online state. If the gateway rejects the new address, it shall go offline.

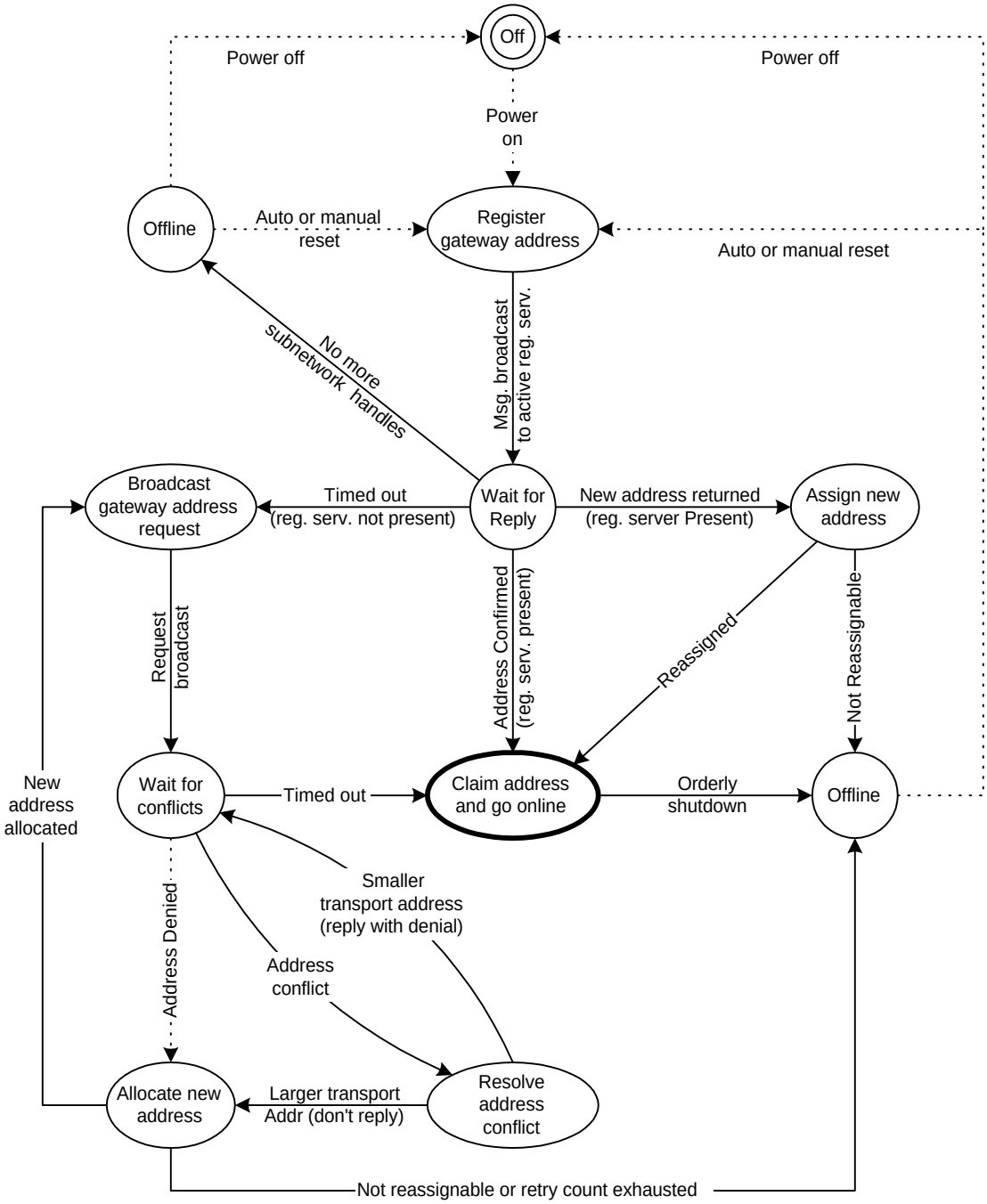

Figure 8 — AES-24 gateway initialization.

2) There is no response from the registry within x_1 seconds. In this case the gateway shall assume a distributed registry and shall proceed by broadcasting its name and handle to all gateway managers on the network. The gateway shall give sufficient time, x_2 seconds, for other gateways to respond. There are three possible outcomes to this:

a) Another gateway responds with a return code indicating that the address has been denied. An address shall be denied for only one of two reasons:

1) the responding gateway is already using the desired address (that is, an address clash);

2) the responding gateway desires the same AES-24 address, and has a numerically smaller transport address.

In either case, the initializing gateway shall either allocate and attempt to register a new address (and proceed as in step 2), or go offline. The gateway may give up (that is, proceed to the offline state) after a certain number of retries (a number that shall be defined by the manufacturer).

b) Another initializing gateway makes a request for the same address. A gateway shall know this by receiving other gateways' address request event messages while waiting during the x_2 time period. In this case, the gateway shall compare its own transport address with that of the other gateway that is requesting the same address to resolve the conflict. There are two possible outcomes to this:

1) Its own transport address is numerically greater than that of the other gateway. In this case, the gateway (not the other gateway) shall either allocate and attempt to register a new address (and proceed as in step 2), or go offline.

2) Its own transport address is numerically smaller than that of the other gateway. In this case, a reply indicating an address deny shall be sent to the other gateway, and the gateway (not the other gateway) shall continue to process more conflicts until the x_2 time period has expired. The other gateway shall follow as in step 2a, above.

c) The x_2 time period expires since neither case 2a nor case 2b1 has been fulfilled. In this case, the gateway shall assume that the requested subnetwork name and subnetwork handle are acceptable and, hence, shall claim the address and proceed to the online state.

Details on gateway initialization messages and required events are considered in AES24-2.

11.4 Subnetwork and internetwork discovery

Subnetwork and internetwork discovery shall be the process of finding out what devices exist in a subnetwork and internetwork, respectively. AES24-2 considers discovery messages for both centralized and distributed registries.

11.5 Inventory change notice

If a registry server is available, an object may monitor changes in the network inventory without the need to poll the network continually. To do so, the object shall send the registry server a request to be placed on its mailing list. When a device manager is registered or deregistered, the registry server shall notify all objects on the mailing list. Details on specific methods and events to perform these actions are considered in AES24-2.

12 Events and control flow

It is common for control and monitoring relationships between objects to have a persistent nature. For example, a particular user interface object that allows the user to control gain will have a relationship with a particular functional object that implements gain control, and this relationship will normally include many control message exchanges and extend over a time period of minutes, hours, or days.

AES-24 events and event tables provide a way to implement such relationships. With appropriately initialized event tables, an AES-24 network may be configured into a distributed event-driven program with a rich set of functions. In such a network, the following kinds of event message exchanges may occur:

a) user control activity triggers events that generate messages to control functional objects;
b) functional object activity triggers events that generate messages to control user interface objects;
c) various software activities (for example, timers) may trigger events that generate messages with a variety of effects.

Although AES-24 messages do not necessarily have to be generated by events, it is anticipated that in typical AES-24 networks, the messages that are not generated directly by events will usually be generated by program processes originally initiated by events.

12.1 Event programming

12.1.1 Event table

Every object that implements one or more events shall have a special property called an event table. This table shall support entries for each kind of event that the object may raise. The event table shall be manipulated through standard event table methods. The internal structure of the event table is not specified; rather, only its functional and interface requirements shall be met. Details are considered in AES24-2.

12.1.2 Event processing

Two event-processing sequences are defined, the closed event processing sequence and the open event processing sequence. An object may use either one.

In the closed sequence, the event message shall be sent with reply requested, and its event table entry shall send no further event messages until the reply is received. The design intent of the closed event processing sequence is to provide a programming mechanism to help prevent cascading events from flooding a network with messages.

In the open event processing sequence, the event message shall be sent with no reply requested, and subsequent event messages may be sent immediately.

12.2 Intermediate objects

Depending on the functions required and the characteristics of the various objects, AES-24 messages passing between objects will sometimes pass through one or more intermediate objects.

Intermediate objects may have various functions, as follows:

a) translating parameter units of measure—sometimes an event message will express a control parameter in units of measure that are different from the units needed by the controlled object, in which case an intermediate object may be interposed to translate;
b) implementing multilateral relationships among controlling and controlled objects, in which several controlling, monitoring, or functional objects may be involved—such relationships occur in control mastering (also known as grouping), indicator mastering, multipoint control, and other cases;
c) implementation of programmed control or monitoring functions, to automate or constrain control and monitoring actions or perform other specialized actions, often of a proprietary nature and possibly involving a large number of other objects.

Figure 9 shows various examples of event message routings and intermediate objects.

The AES-24 standard intermediate object classes are considered in AES24-2. However, it is thought that most intermediate objects will probably implement proprietary functions, and will therefore not be instantiated from the standard classes.

12.3 System builders

To initialize event tables and other operating parameters of network-controlled audio systems, one or more software tools are generally required. AES-24 refers to these tools as system builders.

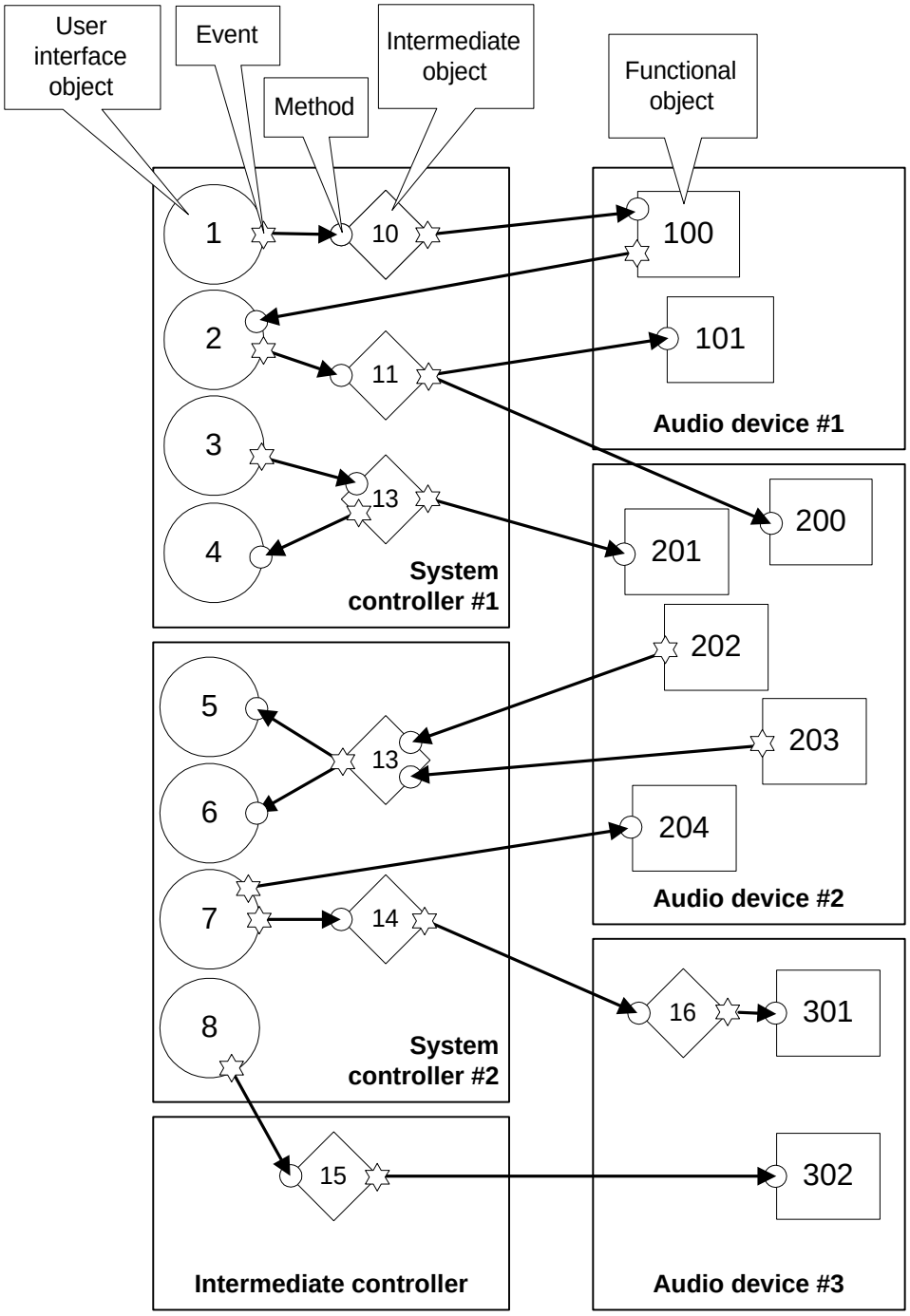

Figure 9 — Event relationships and intermediate objects

In the simplest systems, a system builder might consist only of DIP switches in a small dedicated controller. A system builder of medium complexity may consist of a text editor and a set of text-based system initialization files. An advanced system builder may entail a graphic editor that supports drag-and-drop editing of processing configurations.

In a network with a relatively static configuration, the system builder should be run only when the network is first implemented, or when a piece of networked equipment is replaced. In contrast, a network whose configuration is dynamically changeable should have its system builder running all the time.

12.4 Generic controllers

It is possible to create intelligent system builders that construct whole system controller configurations and sets of control paths automatically or semi-automatically. System controllers that include such system builders shall be referred to as generic controllers.

The term, interrogation, shall denote the process by which objects such as generic controllers acquire in-depth knowledge of devices in the network from the devices themselves.

The specific protocol elements that support interrogation are under consideration, and are not included in the version of AES-24 described in this document.

13 Messages

Every AES-24 message passes from an object to another (or the same) object. All AES-24 messages shall share a common overall format and set of exchange rules. Each AES-24 message shall be an invocation of a specific method in the destination object. The format of the application data contained in the message (the message's parameters) depends on the particular method being invoked.

All messages shall be communicated to and from the transport layer in network byte order, that is, most significant byte first.

In a normal message exchange, an object shall construct and send a message to another object. The message shall identify a target method in the receiving object. Upon receiving the message, the target method shall perform the action indicated by the addressed method.

When the action is complete (or has failed) and if the original message requested a reply, the target method shall return a reply message back to the original object. A reply message shall contain a status code that indicates the outcome of the method's actions.

AES-24 defines a repertoire of generic status-code values. In addition, a class may define method-specific status codes where needed. A flag in the reply message shall indicate which type of status code is being returned. Specific code values are considered in AES24-2.

13.1 Message format

The format of a command or event message in shown in Figure 10.

1 byte	6 byte	6 byte	2 byte	1 byte	variable size
AES_ver	destination	reply	message_id	sequence_no	parameters

Figure 10 — AES-24 command and event message format

The format of a reply message is shown in Figure 11.

1 byte	6 byte	6 byte	2 byte	1 byte	2 byte	variable size
AES_ver	destination	reply	message_id	sequence_no	status_code	parameters

Figure 11 — AES-24 reply message format

The message fields shall be defined in Table 3.

Table 3 — AES-24 message field descriptions

Field name	Description
`AES_ver`	Version of AES24 that is being used by sender; replies shall use the same version
`destination`	Handle to which the message is directed (see 9.2 for format)
`reply`	Handle of the object to which addressed method directs its reply
`message_id`	Specification of message as a command or a reply, and identification of the method that is receiving the command or returning the reply
`sequence_no`	Unique identifier (reply token) that is returned unmodified with a reply message; used to match up replies against requests
`status_code`	Status returned by a method that is replying; field included only with reply messages
`parameters`	Dependant on individual method and in some cases omitted

13.1.1 The `message_id` field

The purposes of the `message_id` field shall be to specify whether a message is a command or a reply message, and to identify the method that is receiving the command or returning the reply.

The `message_id` sub-fields shall be as defined in Table 4.

Table 4 — The `message_id` field

Sub-field name	Bits	Value	Meaning
`type`	15–14	00_2	Command; reply only on error
		01_2	Reply
		10_2	Command; always reply
		11_2	Reserved
`method_level`	13–10	0–15	Level of method in class hierarchy
`method_index`	9–0	0–1023	Index of method within its defining class

13.1.2 The `status_code` field

For a reply message, the `status_code` field shall be as defined in Table 5.

Table 5 — The `status_code` field.

Field	Bits	Value	Meaning
`type`	15	0	Generic status code
		1	Method-specific status code
`status_code`	14–0	0–32767	Status code value

Annex A

(informative)

Simple example system

Figure A.1 shows a hypothetical four-channel distribution amplifier and attached PC-based controller, both of which conform to this standard. The distribution amplifier includes one switch object for turning power on and off, four gain control objects for gain adjustment of the four respective channels, and four level monitoring objects to feed signal level information back to the system controller.

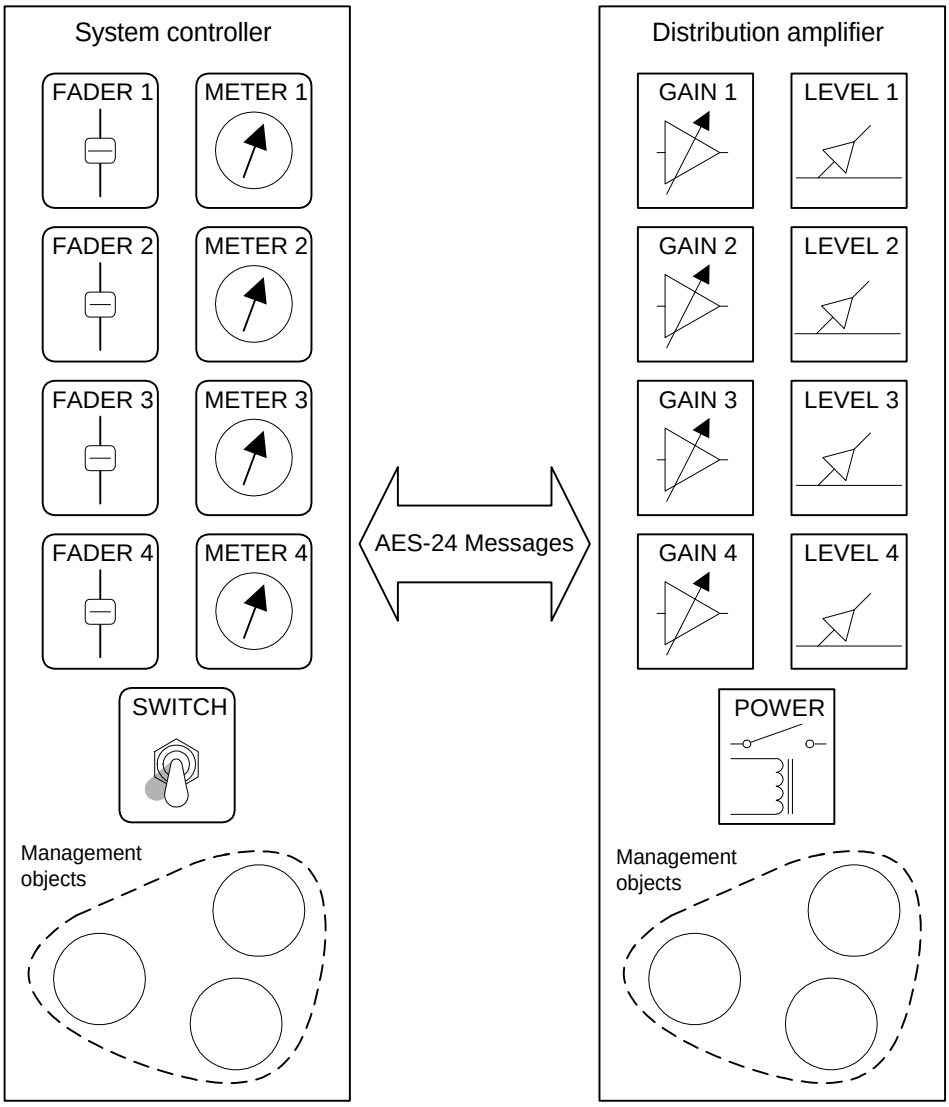

Figure A.1 — Hypothetical system controller and controlled equipment

The PC-based controller for this system runs a graphical computer program whose screen displays four movable fader-like graphics for gain control, four meter-like objects to represent the signal levels, and a switch-like graphic for power on/off control. The standard model considers each of these nine screen elements to be a separate object. Control and monitoring of the distribution amplifier are accomplished by messages that pass back and forth between the controller's screen objects and the distribution amplifier's audio processing objects.

This standard specifies the processes by which these objects identify and learn about each other, and the formats, sequences, and effects of the messages they exchange.

Besides the user interface and audio processing objects, Figure A.1 shows additional objects labeled management objects in both the distribution amplifier and the controller. These objects participate in the operation of the standard protocol but do not contribute directly to electronic or user interface functions. These objects' primary roles are in initialization and configuration processes.

Because this standard is an abstraction, the standard model of the distribution amplifier would be the same regardless of whether the gain control were implemented by four separate physical gain control elements (voltage-controlled amplifiers, motorized potentiometers, and so on) or by digital signal processing software. Similarly, the standard model of the controller would be the same regardless of whether the controller were a graphical PC program or a networkable hardware control surface with physical controls.

Annex B

(informative)

Implementation note: Proprietary central controllers

Although AES-24 generally portrays system controllers as sets of standard user interface objects that control sets of standard functional objects, it also supports simpler network configurations in which the central controller is implemented as one or more large proprietary object(s), and is not subdivided into standard AES objects. Such controllers are characteristic of legacy systems that predate AES-24, but have been adapted to use it. In these cases, the proprietary controllers will usually implement not only user interface functions, but also intermediate object and system builder functions, leading to relatively straightforward sets of control flow relationships, as shown in Figure B.1.

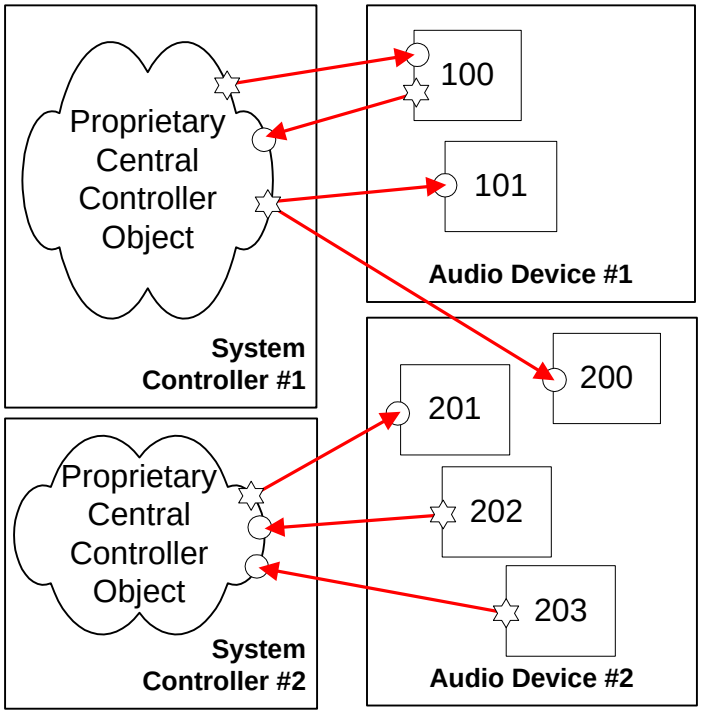

Figure B.1 — Proprietary central controller

Annex C

(informative)

Replacing devices in an initialized network

Following are some ways of handling swapping out of equipment in case of failure.

Strategy 1: Let the system builder do the work.

In this strategy, we assume that we start out with a working system. We assign role name `Main L/R` to a power amp with devicename `DEVXXXX`, and `Side Fill L/R` to a power amp with devicename `DEVYYYY`. If `DEVXXXX` fails, the following sequence occurs:

a) The system builder software alerts the user that `Main L/R` has failed.
b) The user replaces `DEVXXXX` by a new unit, whose devicename is `DEVZZZZ`.
c) The system builder says to the user: we have a new unit `DEVZZZZ`, do you wish to use it as `Main L/R`.
d) The user says yes, and the system builder assigns role name `Main L/R` to `DEVZZZZ`.

Strategy 2: Use the application description field.

In this case, both power amps have sets of DIP switches: `Main L/R` is set to 1, and `Side Fill L/R` is set to 2. These numbers are stored in the application description field. If `Main L/R` fails, the following sequence occurs:

a) The system builder software alerts the user that `Main L/R` has failed.
b) The user goes to the unit and notices that its DIP switches are set to 1. She or he replaces it by a new unit, with its DIP switches set to 1.
c) The system builder software searches for a new unit whose application description field is 1. When it finds it, it assigns the role name `Main L/R` to it and it is ready for use.

Strategy 3: Use a removable serial number device.

This is the same scenario as strategy 1, but the power amps have device names stored in a removable device such as a button cell incorporating a serial number chip. If `DEVXXXX` fails, the following sequence occurs:

a) The system builder software alerts the user that `Main L/R` has failed.
b) The user goes to the rack, removes the button cell from the amp, and puts it in a new amp, which he or she then connects up to the network.
c) The system builder now immediately sees that `DEVXXXX` has come back to life, and tells the user.