An Introductory Tutorial to AES70

Presenters:

Jeff Berryman:
Chair, Technical Committee, OCA Alliance
Chair, SC02-12-L (AES70)
Senior Scientist, Bosch Communications

Simon Jones:
Member, Board of Directors, OCA Alliance
Member, Technical Committee, OCA Alliance
CTO, Focusrite Audio Engineering
AES70 Tutorial Agenda

- Introduction
- Concepts and Overview of the standard
- Object Model
- Connection Management
- Future developments
- Resources
Introduction
Concepts and Overview of the standard
Object Model
Connection Management
Future developments
Resources
You can easily move audio through a network from A to B, but what are you going to do with it when it gets there?
AES70 Tutorial: Introduction

SYSTEM CONTROL

AES70

Legacy Media Protocols
AES67
Dante
AVB
Other IP Media Protocols
WHATEVER
Why another control standard?

- AES70 is the only control architecture that is all of the following:
  - Pro application oriented
  - Scalable up to huge network sizes
  - Suitable for mission-critical applications
  - Friendly to proprietary product features
  - Futureproof
  - Secure
  - Able to provide rich support for reconfigurable devices
  - Heterogeneous-network capable
  - An open public standard
What is “OCA”?

• OCA stands for Open Control Architecture, and is the name of the technology that AES70 standardizes.
  • Compare with MADI, which is standardized in AES10.

• OCA was developed by the OCA Alliance, an open non-profit trade alliance, and given to the AES to standardize.

• The OCA Alliance still exists, and is working hard to promote AES70 adoption and to develop new recommendations for evolving AES70.
• Introduction
• Concepts and Overview of the standard
• Object Model
• Connection Management
• Future developments
• Resources
Basic Elements of AES70

- Specifications
- Class System
- Protocols
The Specification
The Specification

- Divided into three Sections:
The Specification

• Divided into three Sections:
  ○ *AES70-1*: The Framework.
The Specification

- Divided into three Sections:
  - **AES70-1: The Framework.** Defines the models and mechanisms that form AES70.
The Specification

- Divided into three Sections:
  - **AES70-1: The Framework.** Defines the models and mechanisms that form AES70. AES70 is a *control model*, not a *programming model*. 
The Specification

• Divided into three Sections:
  o **AES70-1: The Framework**. Defines the models and mechanisms that form AES70. AES70 is a *control model*, not a *programming model*.
  o **AES70-2: Class Structure**.
The Specification

- Divided into three Sections:
  - **AES70-1: The Framework.** Defines the models and mechanisms that form AES70. AES70 is a *control model*, not a *programming model*.
  - **AES70-2: Class Structure.** Specifies the control class structure, which defines the control and monitoring capabilities of AES70 classes.
The Specification

- **AES70-1: The Framework.** Defines the models and mechanisms that form AES70. AES70 is a *control model*, not a *programming model*.
- **AES70-2: Class Structure.** Specifies the control class structure, which defines the control and monitoring capabilities of AES70 classes.
- **AES70-3: Communication Protocol.**
The Specification

- **AES70-1: The Framework.** Defines the models and mechanisms that form AES70. AES70 is a control model, not a programming model.
- **AES70-2: Class Structure.** Specifies the control class structure, which defines the control and monitoring capabilities of AES70 classes.
- **AES70-3: Communication Protocol.** Defines AES70 remote control and monitoring over a network.
AES70-1: The Framework

- The Open Control Framework, OCF.
- All devices (including controllers) are composed of objects with different object numbers within that device.
- A connection between a controller and a device is called a session, with a **SessionID**, which is unique for the connection between a particular device and controller.
- An object in any device is identified (and addressed) by the unique combination of the SessionID and object number.
AES70 Tutorial: Concepts & Overview

AES70-2: Open Control Class Structure

• Open Control Class Structure, OCC.
• Based on object-orientated programming hierarchical Class methodology.
• Classes are program-code templates for creating objects, in this case controllable and monitorable objects.
• All OCC classes are based on the base class, OcaRoot.
• OcaRoot defines the basic functionality of all OCC class types
• Defines the entire repertoire of objects that an AES70 device can use (Annex A).
• Defines the mandatory objects an AES70 device must implement (Annex B).
OCC is divided into four categories:

• *Workers*
• *Managers*
• *Agents*
• *Networks*
• Workers: Classes that represent signal processing and monitoring functions
• **Workers**: Classes that represent signal processing and monitoring functions

• **Managers**: Classes that represent device housekeeping functions
- **Workers**: Classes that represent signal processing and monitoring functions
- **Managers**: Classes that represent device housekeeping functions
- **Agents**: Classes that represent control-flow processing functions
• Workers: Classes that represent signal processing and monitoring functions
• Managers: Classes that represent device housekeeping functions
• Agents: Classes that represent control-flow processing functions
• Networks: Classes that represent the physical network (or networks) to which the device is connected
Workers are divided into three categories:

- **Actuators** – Signal processing and routing functions
Workers are divided into three categories:

- **Actuators** – Signal processing and routing functions, e.g. gain, mute, source selection
Workers are divided into three categories:

- **Actuators** – Signal processing and routing functions, e.g. gain, mute, source selection
- **Sensors** – Detectors and monitors of various types
• Workers are divided into three categories:
  • Actuators – Signal processing and routing functions, e.g. gain, mute, source selection
  • Sensors – Detectors and monitors of various types, e.g. signal level, gain reduction, temperature
Workers are divided into three categories:

- Actuators – Signal processing and routing functions, e.g. gain, mute, source selection
- Sensors – Detectors and monitors of various types, e.g. signal level, gain reduction, temperature
- Blocks and Matrices – Classes that aggregate objects into structured collections, generally used for modeling / managing complex devices
• Workers are divided into three categories:

  • Actuators – Signal processing and routing functions, e.g. gain, mute, source selection
  • Sensors – Detectors and monitors of various types, e.g. signal level, gain reduction, temperature
  • Blocks and Matrices – Classes that aggregate objects into structured collections, generally used for modeling / managing complex devices, e.g. collecting objects into blocks of “channels”
Signal Processing (Actuators)
- Gain controls
- Mutes
- Switches (n-position)
- Delays
- Equalizers
- Filters (IIR & FIR)
- Limiters & Compressors
- Expanders & Gates
- Levelers
- Signal generators
- Arbitrary numeric and text parameters

Signal Monitoring (Sensors)
- Level sensors (meters)
- Frequency sensors
- Time interval sensors
- Temperature sensors
- Arbitrary numeric sensors
### Basic Actuators
- OcaBooleanActuator
- OcaInt8Actuator, Int16, Int32, Int64
- OcaUint8Actuator, Uint16, Uint32, Uint64
- OcaFloat32Actuator, Float64
- OcaStringActuator
- OcaBitStringActuator

### Basic Sensors
- OcaBooleanSensor
- OcaInt8Sensor, Int16 ...
- OcaUint8Sensor, Int16 ...
- OcaFloat32Sensor, Float64
- OcaStringSensor
- OcaBitStringSensor
Basic Actuators
OcaBooleanActuator
OcaInt8Actuator, Int16, Int32, Int64
OcaUint8Actuator, Uint16, Uint32, Uint64
OcaFloat32Actuator, Float64
OcaStringActuator
OcaBitStringActuator

Basic Sensors
• OcaBooleanSensor
• OcaInt8Sensor, Int16 ...
• OcaUint8Sensor, Int16 ...
• OcaFloat32Sensor, Float64
• OcaStringSensor
• OcaBitStringSensor

+ Proprietary extensions as needed
AES70 Tutorial: Concepts & Overview

AES70-2: Non-standard Classes

- Also termed “proprietary” classes
- Follow the same rules as the OCC class tree
- They are an extension [derivative] of a standard class
  - Only derived from a single standard class
  - Must include the same functionality as the derivative standard class
  - Enhance the definitions of existing features
  - Can have extra functionality and features beyond the standard class
AES70-2: Non-standard Classes

• OCC Derivation Example:
• OcaSwitch (1.1.1.4)
AES70 Tutorial: Concepts & Overview

AES70-2 : Non-standard Classes

- OCC Derivation Example:
- OcaSwitch (1.1.1.4)
- OcaActuator (1.1.1)
AES70-2 : Non-standard Classes

• OCC Derivation Example:
  - OcaSwitch (1.1.1.4)
  - OcaActuator (1.1.1)
  - OcaWorker (1.1)
AES70-2 : Non-standard Classes

- OCC Derivation Example:
  - OcaSwitch (1.1.1.4)
  - OcaActuator (1.1)
    - OcaWorker (1.1)
  - OcaRoot (1)
AES70-2 : Non-standard Classes

• OCC Derivation Example:
  • OcaSwitch (1.1.1.4)
  • OcaActuator (1.1.1)
    • OcaWorker (1.1)
    • OcaRoot (1)
  • OcaSwitchAES (1.1.1.4.[MfrID].1):
    • *Has all the features of the OcaSwitch*
    • *Plus the “extra” functionality required*
AES70 Tutorial: Concepts & Overview

AES70-2, Mandatory Objects (Annex B):
AES70-2, Mandatory Objects (Annex B):

- Mandatory objects have defined (fixed) object numbers (oNo).
AES70-2, Mandatory Objects (Annex B):

- Two Managers

  1. OcaDeviceManager (oNo:1) – Overall device manager, containing the Device Name, Manufacturer Name, Serial Number and ModelGUID etc.
AES70-2, Mandatory Objects (Annex B):

- Two Managers
  1. *OcaDeviceManager* (oNo: 1)
  2. *OcaSubscriptionManager* (oNo: 5) – Manages reporting of device data back to controllers.
AES70-2, Mandatory Objects (Annex B):

- Two Managers

1. `OcaDeviceManager (oNo:1)`
2. `OcaSubscriptionManager (oNo:5)` – Manages reporting of device data back to controllers. Not actually mandatory, but its absence would imply a polled system, which may be okay for small devices.
AES70-2, Mandatory Objects (Annex B):

• One Worker

  1. OcaBlock (oNo:100) – The “root” block, which contains all the device’s worker objects
AES70 Tutorial: Concepts & Overview

AES70-3: Protocol for IP Networks

• Referred to as OCP.1.
• AES70 only uses *standard* transport protocols.
• Devices are “discovered” by interested controllers using DNS-SD service discovery.
• DNS-SD is often referred to by its common implementation, *Bonjour*
AES70-3: Protocol for IP Networks

• Currently, Four Supported Services:
  1. TCP/IP (_oca._tcp)
  2. UDP (_oca._udp)
  3. Web-socket (_ocaws._tcp)
  4. Secure via Pre-shared Key (_ocasec._tcp)

• At least one service must be supported
AES70 Network Traffic

**CONTROLLER**

- **COMMAND**
  - Object number: <nn>,
  - Parameter number: <mm>,
  - Set value to: <xx>

- **RESPONSE**
  - Status = done ok, failed, etc.

- **NOTIFICATION**
  - Object number: <nn>,
  - Event <kk> occurred

**DEVICE**

- Object

Connected together by the unique SessionID
AES70 Tutorial Agenda

• Introduction
• Concepts and Overview of the standard
• Object Model
• Connection Management
• Future developments
• Resources
AES70 Objects / Classes

- **Classes** are comprised of three different features:
  - **Variables**: Stores that determine the classes configuration
  - **Properties**: Convenient ways of either retrieving a variables’ state (a getter), or changing a variable state (a setter)
  - **Methods**: Functions of the class, also referred to as Member Functions

- **All AES70 Classes** are derived from the **OcaRoot class**
Derived Classes

• An analogy:
  • *The Russian Doll...*
Derived Classes

• An analogy:
  • The Russian Doll...
## OcaRoot

### Control Classes: OcaRoot

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>ClassID : OcaClassID = 1</td>
<td>{readOnly}</td>
</tr>
<tr>
<td>ClassVersion : OcaClassVersionNumber = 1</td>
<td>{readOnly}</td>
</tr>
<tr>
<td>ObjectNumber : OcaONo</td>
<td>{readOnly}</td>
</tr>
<tr>
<td>Lockable : OcaBoolean</td>
<td>{readOnly}</td>
</tr>
<tr>
<td>Role : OcaString</td>
<td>{readOnly}</td>
</tr>
<tr>
<td>PropertyChanged(OcaPropertyChanged)</td>
<td></td>
</tr>
<tr>
<td>GetClassIdentification(OcaClassID*)</td>
<td>OcaStatus</td>
</tr>
<tr>
<td>GetLockable(OcaBoolean*)</td>
<td>OcaStatus</td>
</tr>
<tr>
<td>Lock()</td>
<td>OcaStatus</td>
</tr>
<tr>
<td>Unlock()</td>
<td>OcaStatus</td>
</tr>
<tr>
<td>GetRole(OcaString*)</td>
<td>OcaStatus</td>
</tr>
</tbody>
</table>
OcaRoot Immediate Derivatives

**Control Classes::OcaRoot**

- ClassID : OcaClassID = 1 (readOnly)
- ClassVersion : OcaClassVersionNumber = 1 (readOnly)
- ObjectNumber : OcaONo (readOnly)
- Lockable : OcaBoolean (readOnly)
- Role : OcaString (readOnly)

**Workers::OcaWorker**

- ClassID : OcaClassID = 1.1 (readOnly)
- ClassVersion : OcaClassVersionNumber = 1 (readOnly)
- Enabled : OcaBoolean
- Ports : OcaList<OcaPort>
- Label : OcaString
- Owner : OcaONo
- Latency : OcaTimeInterval

**Agents::OcaAgent**

- ClassID : OcaClassID = 1.2 (readOnly)
- ClassVersion : OcaClassVersionNumber = 1 (readOnly)
- Label : OcaString
- Owner : OcaONo

**Managers::OcaManager**

- ClassID : OcaClassID = 1.3 (readOnly)
- ClassVersion : OcaClassVersionNumber = 1 (readOnly)
OcaSensor Derivatives

**Object Model**

**class Sensor Classes**

- **abstract**
  - Workers::OcaWorker

- Sensors::OcaSensor
  - ClassID: OcaClassID = 1.1.2.6 (readOnly)
  - ClassVersion: OcaClassVersionNumber = 1 (readOnly)
  - Identify(OcaEventData) : void
  - GetReadingState(OcaSensorReadingstate*) : OcaStatus

- Sensors::OcaAudioLevelSensor
  - ClassID: OcaClassID = 1.1.2.2.2 (leaf)
  - Law: OcaLevelMeterLaw
  - GetLaw(OcaLevelMeterLaw*) : OcaStatus
  - SetLaw(OcaLevelMeterLaw) : OcaStatus

- Sensors::OcaIdentificationSensor
  - ClassID: OcaClassID = 1.1.2.6 (readOnly)
  - ClassVersion: OcaClassVersionNumber = 1 (readOnly)
  - Identify(OcaEventData) : void

- Sensors::OcaTemperatureSensor
  - ClassID: OcaClassID = 1.1.2.5 (readOnly)
  - ClassVersion: OcaClassVersionNumber = 1
  - Reading: OcaTemperature
  - GetReading(OcaTemperature*, OcaTemperature*, OcaTemperature*): OcaStatus

- Sensors::OcaTimeIntervalSensor
  - ClassID: OcaClassID = 1.1.2.3 (readOnly)
  - ClassVersion: OcaClassVersionNumber = 1
  - Reading: OcaTimeInterval
  - GetReading(OcaTimeInterval*, OcaTimeInterval*, OcaTimeInterval*) : OcaStatus

- Sensors::OcaFrequencySensor
  - ClassID: OcaClassID = 1.1.2.4 (readOnly)
  - ClassVersion: OcaClassVersionNumber = 1
  - Reading: OcaFrequency
  - GetReading(OcaFrequency*, OcaFrequency*, OcaFrequency*) : OcaStatus
OcaSensor Derivatives

- In doll terms:

  - OcaAudioLevelSensor
  - OcaLevelSensor
  - OcaSensor
  - OcaWorker
  - OcaRoot
AES70 Tutorial Agenda

- Introduction
- Concepts and Overview of the standard
- Object Model
- Connection Management
- Future developments
- Resources
AES70 Connection Management (AES70 CM3)

- Set of classes defined in the AES70 Object Model that support the use of AES70 for managing stream data connections
- CM3 allows a standard method for controllers to make or break streamed audio paths.
- Adaptable to many streaming protocols - AES67, SMPTE ST 2100-30, Dante, AVB, etc.
- Not a protocol converter - does not convert between streams of different media transport protocols
AES70 Tutorial: Connection Management

AES70 CM3 for AES67

• Why?
  • AES67 specifies the audio streaming protocols, methods and requirements
  • It does not specify connection management. Instead, it offers the option of implementing any of four existing connection management methods
  • A common connection management scheme is needed for interoperability

• CM3 offers a common method of connection management for all AES67 products
• CM3 can work alongside of existing connection management implementations to provide a single interoperable connection management mechanism
AES70 CM3 for AES67

- An AES67 CM3 adaptation is in the process of being ratified by the AES70 standards committee SC-02-12-L.
- The CM3 adaptation is a recommended practice for use of the current AES70-2018 specification.
- Future adaptations will be published in the same form as the AES67 adaptation. These will include:
  - *SMPTE ST 2110-30*
  - *Dante*
  - *AVB (Milan)*
• Introduction
• Concepts and Overview of the standard
• Object Model
• Connection Management
• Future developments
• Resources
JSON encoding of OCP.1 Protocol

- OCP.1 is a binary protocol
- Web-based programmers prefer text-based protocols.
- A JSON encoding of OCP.1 has been developed.
  
  It will be part of the next revision of AES70, or possibly as an amendment to AES70-2018

- Status:
  
  - The specification is complete
  - Discussions are ongoing about how to deal with interoperability issues that arise when multiple encodings are available
• Introduction
• Concepts and Overview of the standard
• Object Model
• Connection Management
• Future developments
• Resources
AES70 Resources

• AES70-2018, AES Standards
• OCA Alliance web-site: https://www.ocaalliance.com/
• OCA Alliance Technical Site: https://ocaalliance.github.io/index.html
• AES70 enabled products: https://www.ocaalliance.com/aes-70-products/