AES147: PD16

Adding AES70 Control to Your Device
PD16 Making an AES70 Device:

Presenters:

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

Ethan Wetzell:
Chair, Marketing Committee, OCA Alliance
Platform Strategist, Bosch Communications
Making an AES70 Device: Agenda

- Device Concepts
- Controller Basics
- Device Implementation
- Resources
- Demonstrations
Making an AES70 Device: Agenda

• Device Concepts
• Controller Basics
• Device Implementation
• Resources
• Demonstrations
Making an AES70 Device: Concepts

Basic Elements of AES70

- Specifications
- Class System
- Protocols
Making an AES70 Device: Concepts

The Specification
Making an AES70 Device: Concepts

The Specification

- Divided into three Sections:
The Specification

- Divided into three Sections:
  - **AES70-1**: The Framework.
Making an AES70 Device: Concepts

The Specification

• Divided into three Sections:
  o **AES70-1: The Framework.** Defines the models and mechanisms that form AES70.
Making an AES70 Device: Concepts

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.
Making an AES70 Device: Concepts

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.**
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.
Making an AES70 Device: Concepts

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.
  - **AES70-3: Communication Protocol.**
Making an AES70 Device: Concepts

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.
  - **AES70-3: Communication Protocol.** Defines AES70 remote control and monitoring over a network.
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).
Making an AES70 Device: Concepts

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
Making an AES70 Device: Concepts

- **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

Making an AES70 Device: Concepts
• 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”*
Making an AES70 Device: Concepts

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
Making an AES70 Device: Concepts

**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
AES70-2 : 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 have 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)
Making an AES70 Device: Concepts

AES70-2: Non-standard Classes

- OCC Derivation Example:
- OcaSwitch (1.1.1.4)
- OcaActuator (1.1.1)
Making an AES70 Device: Concepts

AES70-2 : Non-standard Classes

- OCC Derivation Example:
- OcaSwitch (1.1.1.4)
- OcaActuator (1.1.1)
  - OcaWorker (1.1)
Making an AES70 Device: Concepts

AES70-2 : Non-standard Classes

- OCC Derivation Example:
- OcaSwitch (1.1.1.4)
- OcaActuator (1.1.1)
  - 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
AES70-2, 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
AES70-3: 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*
Making an AES70 Device: Concepts

AES70-3: Protocol for IP Networks

• Up to 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
Making an AES70 Device: Agenda

- Device Concepts
- **Controller Basics**
- Device Implementation
- Resources
- Demonstrations
Selected Concepts for AES70 Controllers

• DNS-SD, registration, browsing etc.
• Subscriptions
• Enumeration and name searching
• AES70 and User Interfaces
• Keeping the connection and knowing when it is gone
• Security
Making an AES70 Device: Controllers

DNS-SD, registration, browsing etc.

- Formerly known as Bonjour
- AES70 is using a registered service identifier
- A device registers itself with the service (multicast, here I am)
- Controllers can browse registrations, get an inventory of devices
- The registration will contain IP address and port number
- The controller can make a connection to the device.
- Through the Device object and other required objects, the controller can obtain detailed information about the device.
Subscriptions

• Keeping Controllers up to date (without periodic polling)
• Subscribe to all or selected device objects
• Get notified changes when they occur
• Subscribing to sensors
• Observers
Enumeration and Name Searching

• The device object can be enumerated
• Depth-first walk of all objects in a device
• Objects can also be found through searching by name
• Includes wildcards, fully recursive
AES70 and User Interfaces

- AES70 purposely does not specify UI aspects
- It does allow for logical grouping of objects which can direct GUI generation
- The strong typing enables automatic mapping to GUI objects
- There are no controller classes, controllers control device objects
- An example of a controller implementation could create a proxy class instance for each object in a device
- Example, execute SetGain(v) on a proxy object will effectively result in that method being executed in the device using the protocol
Keeping the connection and knowing when it is gone

• Keep alive mechanism on top of what the transport layer might offer
• Allows controller and device to know when a session is done
• Both ends agree on a keep alive interval
• If no communication has been seen in that interval the session will end
• Improves connection loss detection on both connection and connectionless transport layers
• When a session is terminated all state related to the connection is freed

• Reset device command, can be broadcast
Security

- AES70 offers secure operation on the network as an option (TLS)
- Installation of pre-shared keys is application specific
- Access control is not part of AES70
- A trusted controller can implement access control in an application specific manner
Making an AES70 Device: Agenda

- Device Concepts
- Controller Basics
- Device Implementation
- Resources
- Demonstrations
Making an AES70 Device: Devices

Implementation of AES70 Devices
Three examples
Implementation of AES70 Devices, three examples:

- A commercial audio device
- Demo Devices:
  - Simple non-audio control device
  - Development audio streaming device
Making an AES70 Device: Audio Device

A Commercial Implementation of AES70:
Focusrite RedNet4:

- Eight Channel Microphone Preamplifier
- IP audio product based on Audinate Dante / AES67
- Remote controlled via the IP network using a Focusrite proprietary protocol
- RedNet range is an ideal target for AES70
Implementation Requirements:

• Must be a simple firmware upgrade, no hardware changes
• Must be backwards compatible so that existing control software remains fully functional
The Proxy Solution:

- Allow the existing firmware paradigm to continue to function, giving backwards compatibility
- Act as a “bridge” between the existing control protocol and AES70
Making an AES70 Device: Audio Device

Diagram showing the structure of AES70 and RedNet devices, including RedNet Control, TCP/IP Network, RedNet Device (RednetOCA and RednetOCA), and RedUnit.
Making an AES70 Device: Audio Device
Proxy solution. Developed on a Windows PC, with a simple port to the target: Very Efficient and Quick
The Proxy Solution:

- Allow the existing firmware paradigm to continue to function, giving backwards compatibility
- Act as a “bridge” between the existing control protocol and AES70
- Allows for the creation of virtual devices
- Virtual devices allow ecosystem development without needing multiple hardware units
AES70 Channel Control Classes:
AES70 Channel Control Classes: Channels 1 & 2
AES70 Channel Control Classes: Channels 3 - 8
AES70 Media Networking Related Classes

Control of the Device Sample Rate

The Streaming Network: Dante adaption derived class

Advertisement of the streaming source audio channels
Implementation of AES70 Devices, two examples:

- An audio device
- A simple non-audio control device
Making an AES70 Device: Non-Audio Device

The OCA Microdemo

120 mm

165 mm
Brief:

• To show that it’s possible to implement OCA in small embedded processor environments
Hardware Overview:

• CPU: ST Microelectronics STM32F207VET6 (512kB flash, 128kB SRAM, 120MHz Cortex M3)
• 10/100 baseT Ethernet
• Eight switches with LED’s
• Two rotary encoders
• Eight LED bargraph meters, six segments
• Two GPO outputs, controlling relays for isolated control
• USB 2.0 full speed, for future use.
AES70 Implementation:

• *Make use of the repertoire of “simple” actuators and sensors, keeping it as generic as possible*

• *Only four different worker classes required for all functions*
Making an AES70 Device: Non-Audio Device

AES70 Objects:

- Switches with LED's
- Relays
- Encoders
- Audio Level Meters
AES70 Objects:

OcaBitStringActuator (8W)
AES70 Objects:

- OcaBitStringActuator (8W)
- OcaBitStringSensor (8W)
AES70 Objects:

- OcaBitStringActuator (8W)
- OcaBitStringSensor (8W)
- 2 of OcaInt8Sensor (-128 to 127 with wrap)
AES70 Objects:

- OcaBitStringActuator (8W)
- OcaBitStringSensor (8W)
- 2 of OcaInt8Sensor (-128 to 127 with wrap)
- 2 of OcaBooleanActuator
AES70 Objects:

- OcaBitStringActuator (8W)
- OcaBitStringSensor (8W)
- 2 of OcaInt8Sensor (-128 to 127 with wrap)
- 2 of OcaBooleanActuator
- 8 of OcaBitStringActuator (6W)
Core Firmware Implementation:

<table>
<thead>
<tr>
<th>DNS-SD</th>
<th>OCA Stack</th>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>lwIP (1.4.1)</td>
<td>FreeRTOS (7.1.1)</td>
<td></td>
</tr>
</tbody>
</table>

Hardware Abstraction Layer

OCA MicroDemo Hardware

The firmware architecture is straightforward and familiar to many embedded developers.
Implementation of AES70 Devices, two examples:

- An audio device
- A simple non-audio control device
- A development streaming device
Making an AES70 Device: Audio Device

RedNet Desktop Functional Block Diagram

- RJ45
- RJ45
- RJ45
- RJ45
- RJ45

Ethernet Switch

Brooklyn Module

ADC

DAC

DAC

Attenuator

Attenuator

Line In

Line Out

Headphones
Device Manager and Identification:

<table>
<thead>
<tr>
<th>ModelDescription</th>
<th>Focusrite</th>
<th>PTTest</th>
<th>0.01</th>
</tr>
</thead>
<tbody>
<tr>
<td>OcaVersion</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>State</td>
<td>Operational</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Serial</td>
<td>00130e000122</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ModelGUID</td>
<td>00130E</td>
<td>00010001</td>
<td></td>
</tr>
</tbody>
</table>
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/
AES70 Control: Demo Setup

Browsing and Enumeration

Virtual Device
Desktop Device
Microdemo Device
AES70-UDP Device

Wireless Browser
AES70 Objects:

OcaBitStringActuator (8W)
OcaBitStringSensor (8W)

2 of OcaInt8Sensor (-128 to 127 with wrap)

2 of OcaBooleanActuator
8 of OcaBitStringActuator (6W)
Making an AES70 Device: Audio Device
Controllers

AoIP Source  →  Desktop Device  →  Microdemo Device  →  Controller

Wireless: Controller  Java Server