Media

#WWDC14

What’s New in Core Audio

Session 501

Torrey Holbrook Walker
Master of Ceremonies

What's New in Core Audio

Overview

Core MIDI enhancements

Inter-App Audio UI views

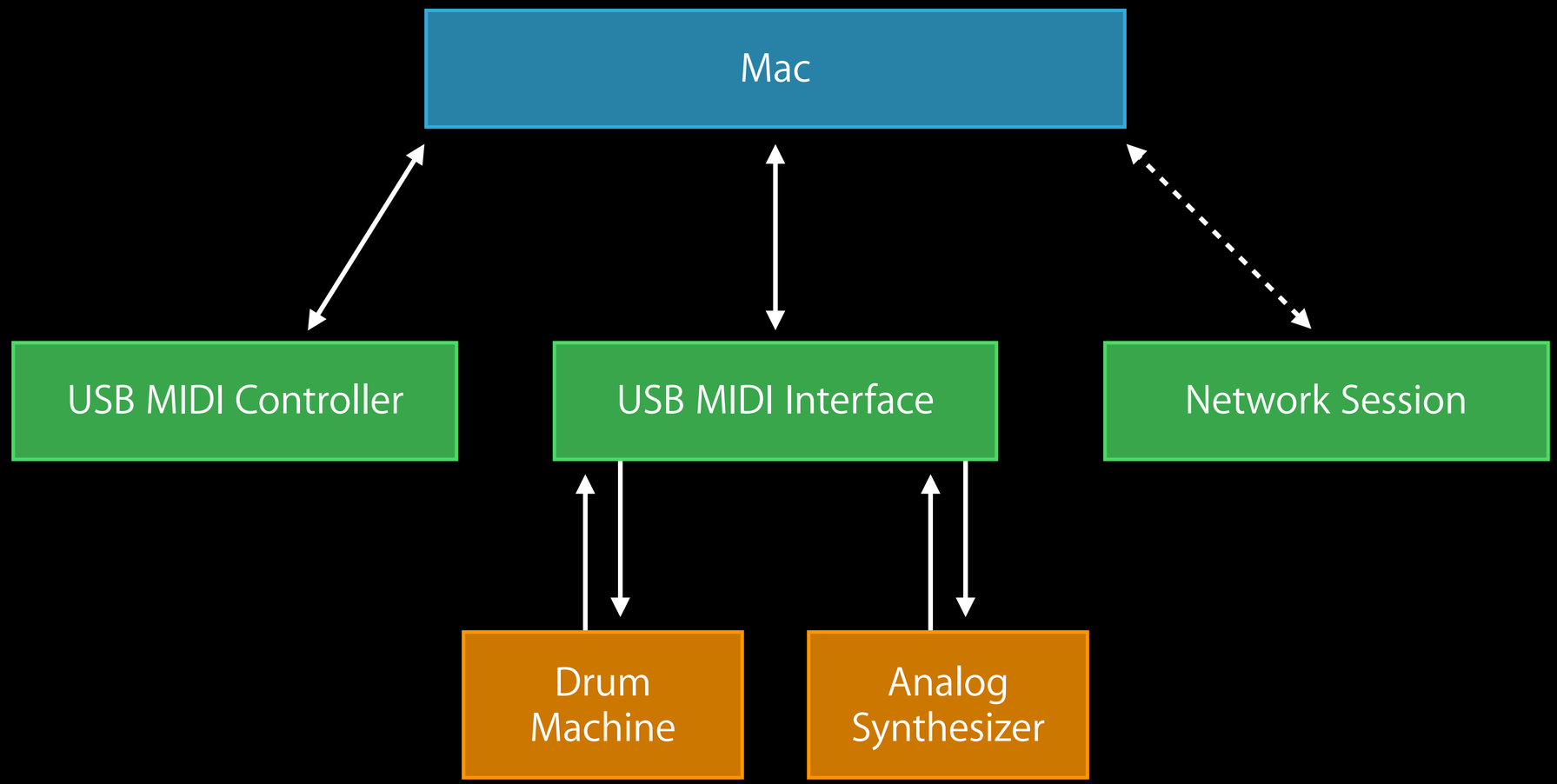
Enhanced AV Foundation audio

- Audio Unit Manager
- AVAudioSession
- Utility classes
- AVAudioEngine

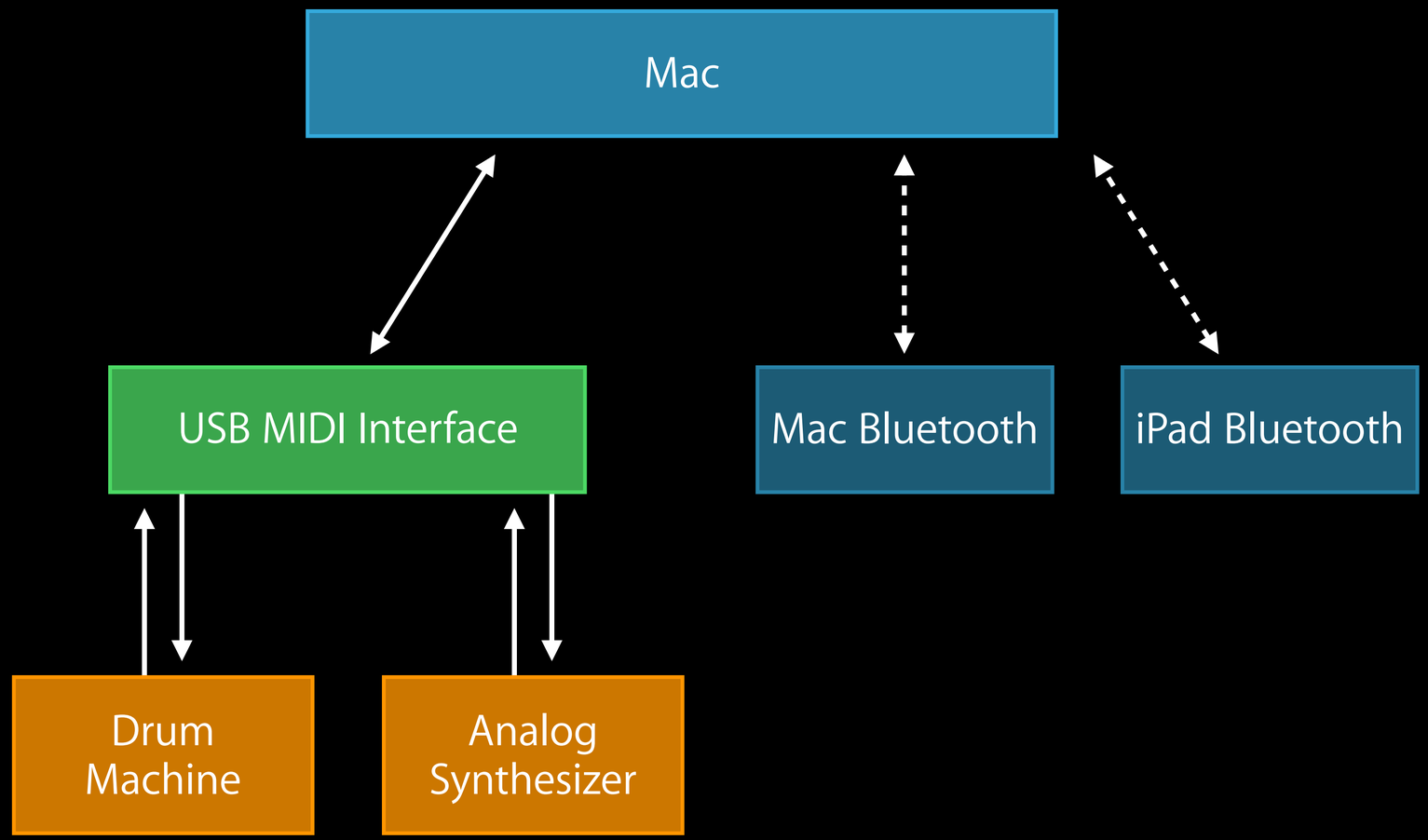
What's New with Core MIDI

Torrey Holbrook Walker
Vice Chair, Itty Bitty MIDI Committee

Your Current MIDI Studio



Your New MIDI Studio



MIDI over Bluetooth

Send and receive MIDI over Bluetooth LE for OS X and iOS

Connection is secure

Appears as an ordinary MIDI device

Bluetooth Overview

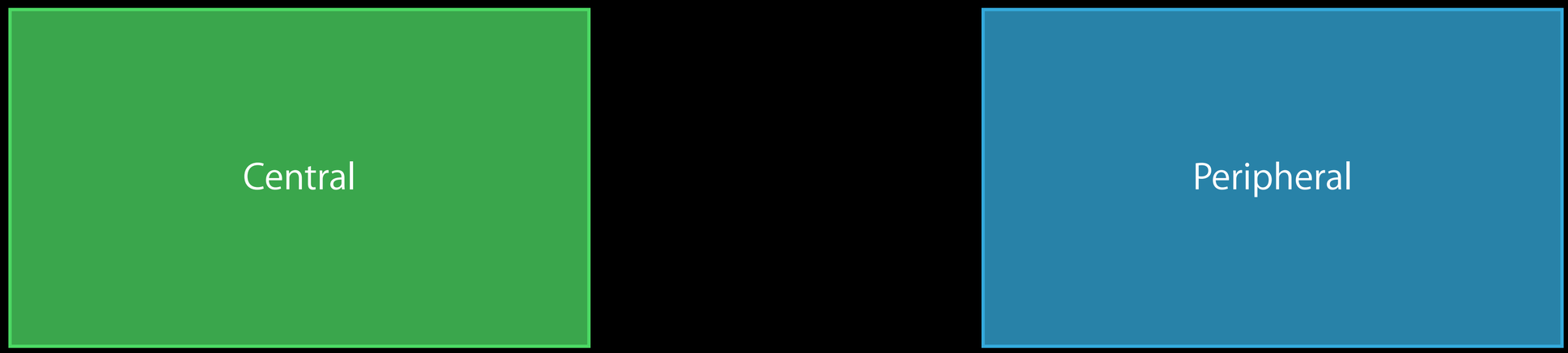
Two key roles

Central

Peripheral

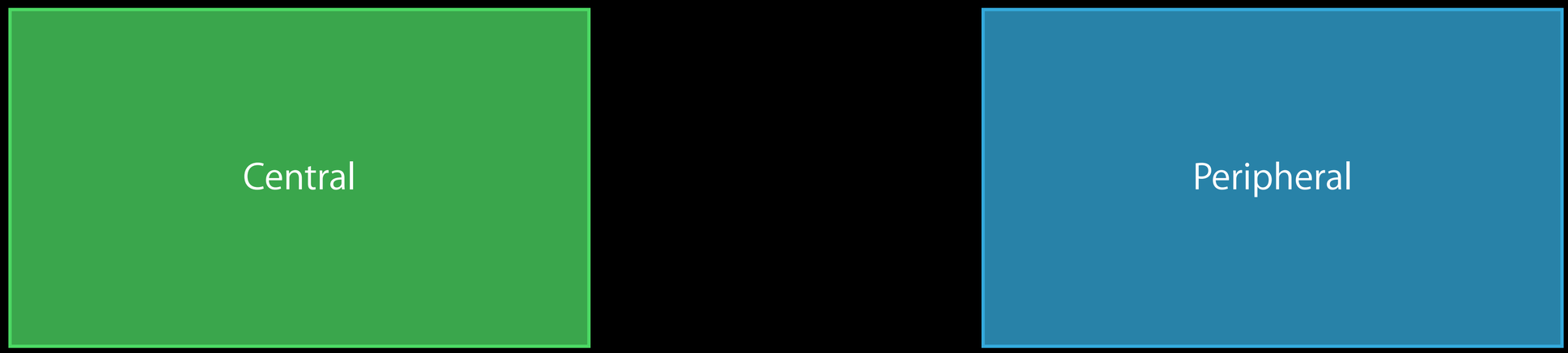
Bluetooth Overview

Peripheral advertises MIDI capabilities



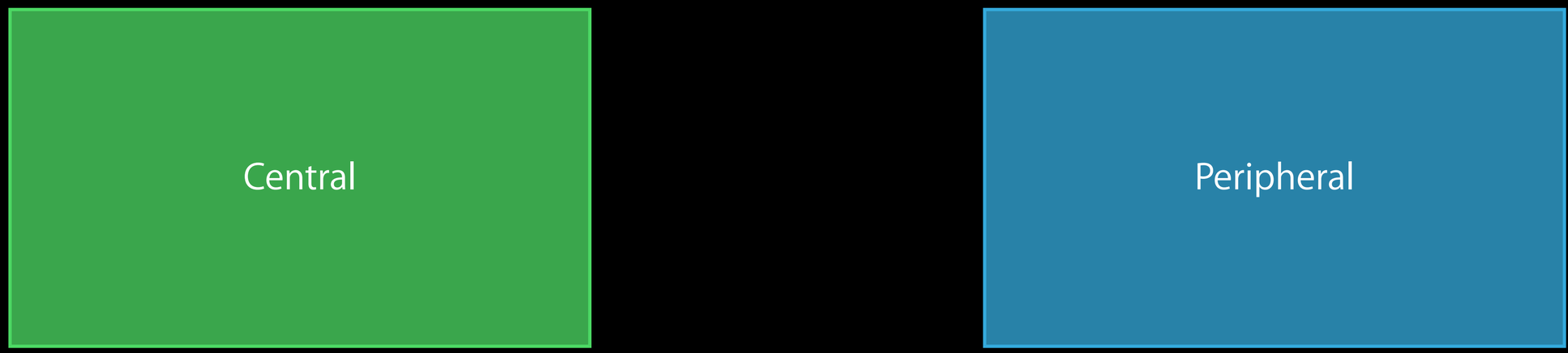
Bluetooth Overview

Peripheral advertises MIDI capabilities



Bluetooth Overview

Central scans and connects

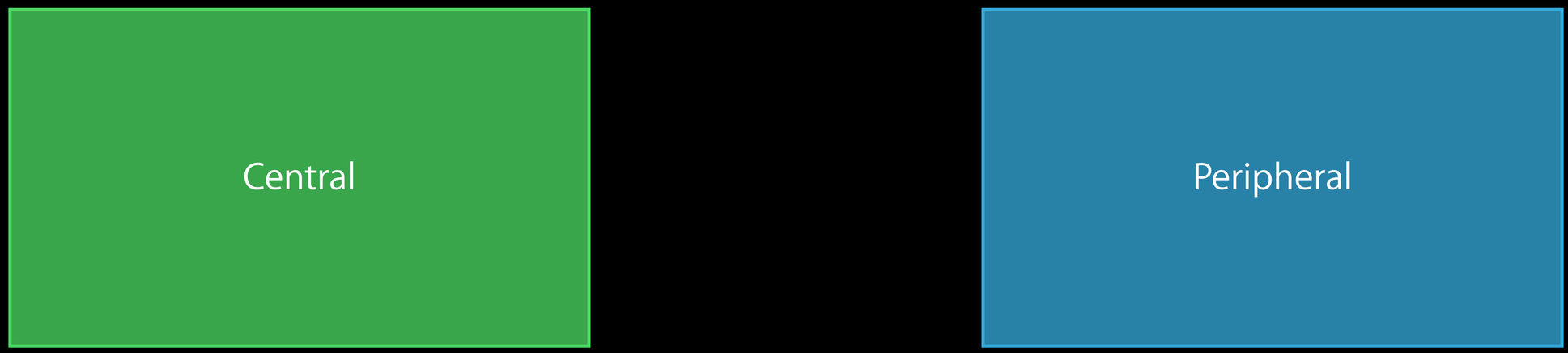


Bluetooth Overview

Central scans and connects

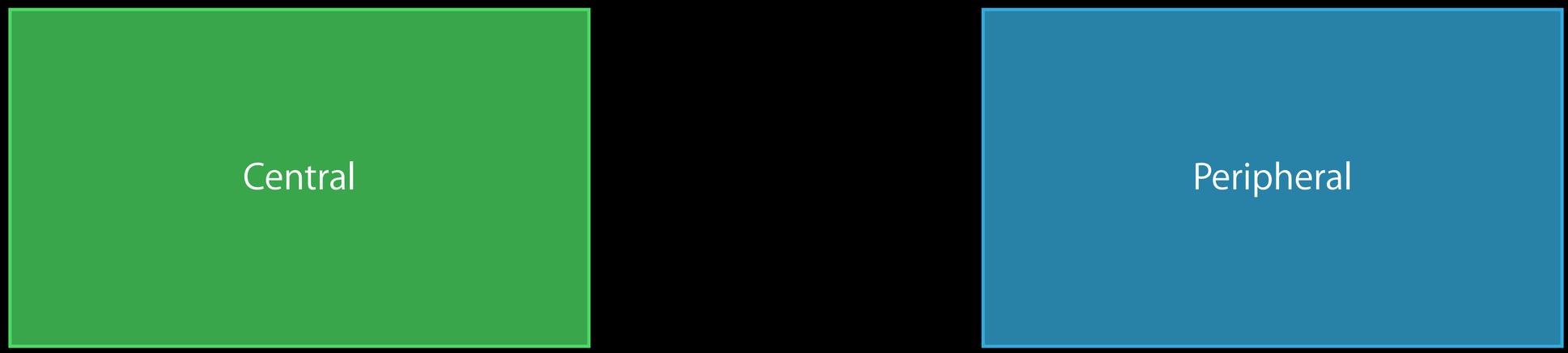
Bluetooth Overview

Bluetooth MIDI I/O via characteristic



Bluetooth Overview

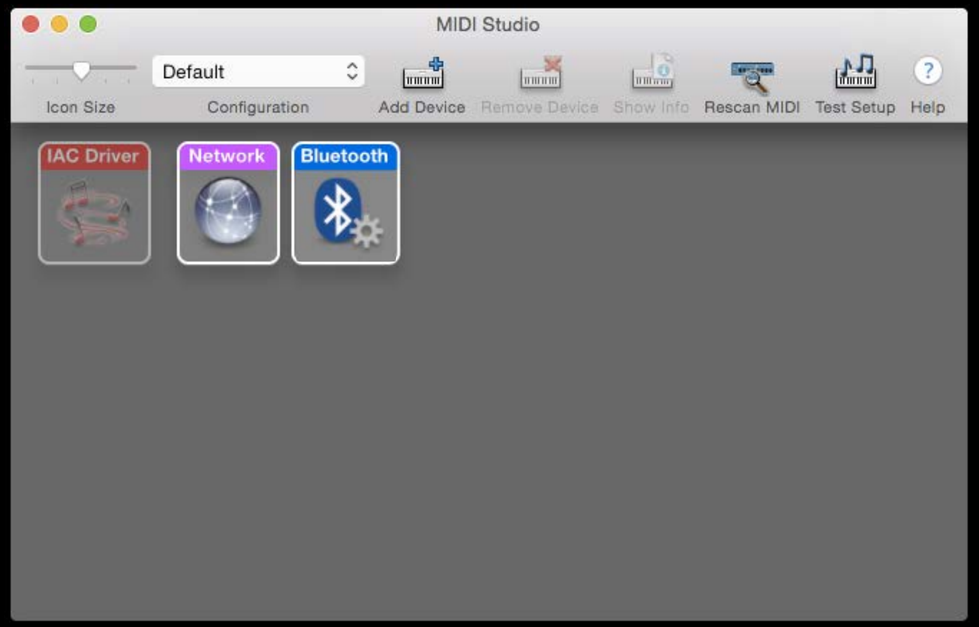
Bluetooth MIDI I/O via characteristic



OS X—You're Ready Already

Audio MIDI Setup

- New Bluetooth panel

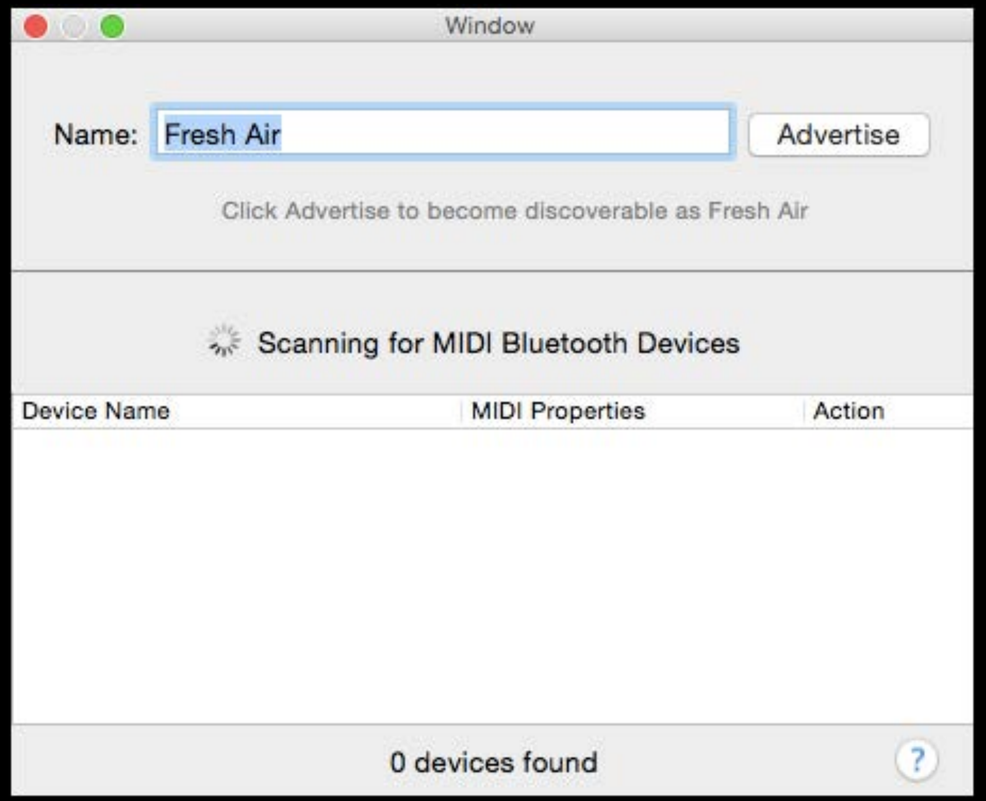


OS X—You're Ready Already

Advertise

Scan/connect

Creates standard MIDI devices



iOS—CoreAudioKit View Controllers

New UI for Bluetooth MIDI

Required to manage Bluetooth connections for your app

“Scan” or “Advertise”

Unused connections are terminated after a few minutes

Demo

Bluetooth MIDI UI on Mac OS X + iOS

Final Thoughts

Works with Mac, iPhone and iPad with native Bluetooth LE support

Low latency

Bluetooth LE bandwidth >> 3,125 B/s

Standardization in the works

Add Bluetooth UI views on iOS

CoreAudioKit Framework

Michael Hopkins
Core Audio User Interface Wizard

What Is CoreAudioKit?

New iOS framework

Provides standardized user interface elements

- MIDI over Bluetooth LE
- Inter-App Audio views

Easy to adopt

Works on iPhone and iPad

MIDI over Bluetooth LE

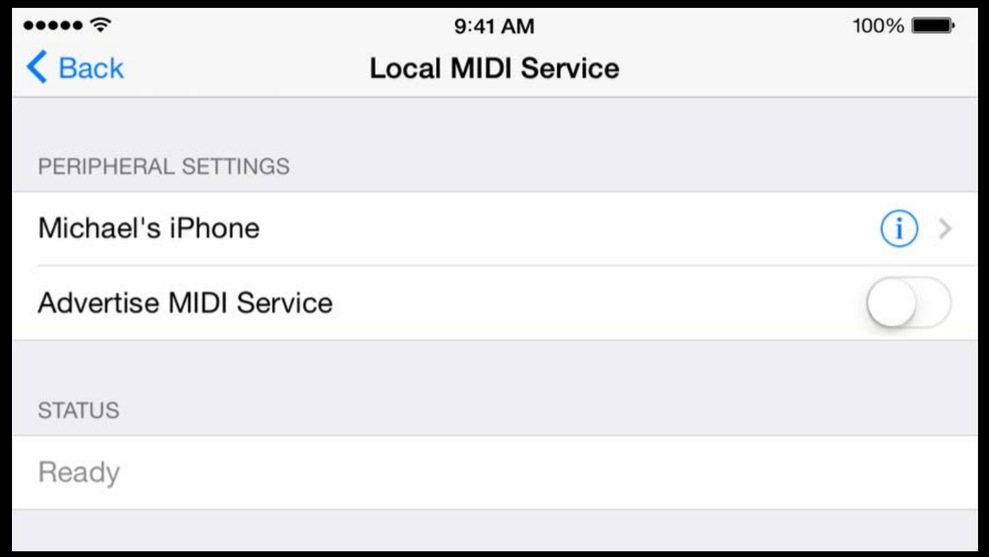
User interface elements

Michael Hopkins
Core Audio User Interface Wizard

MIDI over Bluetooth LE

CABTMIDILocalPeripheralViewController

Required to advertise iOS device as a Bluetooth peripheral



Using BLE UI Classes

```
#import <CoreAudioKit/CoreAudioKit.h>
```

Configuring a local peripheral

```
CABTMIDILocalPeripheralViewController *viewController =
    [[CABTMIDILocalPeripheralViewController alloc] init];
[self.navigationController pushViewController: viewController animated: YES];
```

Using BLE UI Classes

```
#import <CoreAudioKit/CoreAudioKit.h>
```

Configuring a local peripheral

```
CABTMIDILocalPeripheralViewController *viewController =
    [[CABTMIDILocalPeripheralViewController alloc] init];
[self.navigationController pushViewController: viewController animated: YES];
```

Using BLE UI Classes

```
#import <CoreAudioKit/CoreAudioKit.h>
```

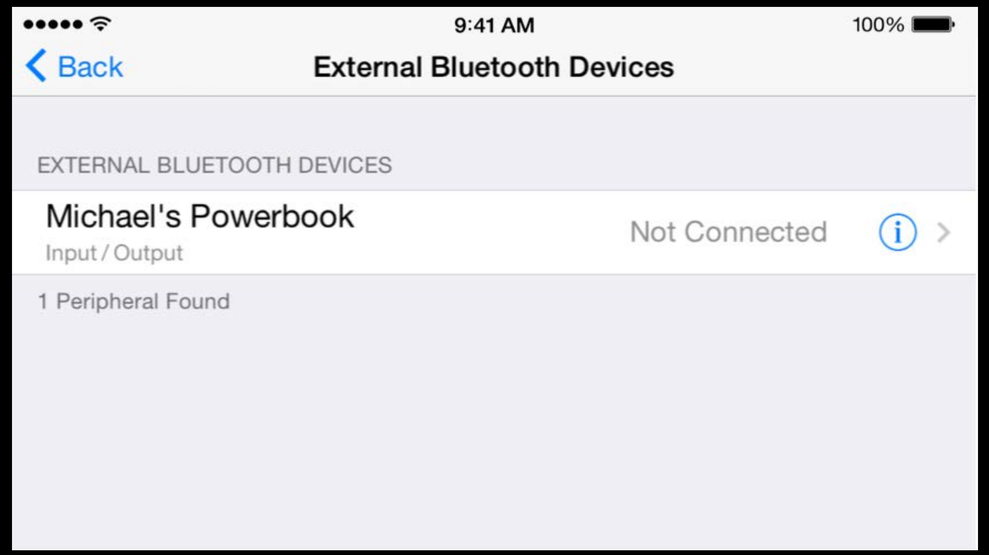
Configuring a local peripheral

```
CABTMIDILocalPeripheralViewController *viewController =
    [[CABTMIDILocalPeripheralViewController alloc] init];
[self.navigationController pushViewController: viewController animated: YES];
```

MIDI over Bluetooth LE

CABTMIDICentralViewController

Required for discovering and connecting to Bluetooth peripherals



Using BLE UI Classes

```
#import <CoreAudioKit/CoreAudioKit.h>
```

Searching for and connecting to external devices

```
CABTMIDICentralViewController *viewController =
    [[CABTMIDICentralViewController alloc] init];
[self.navigationController pushViewController: viewController animated: YES];
```

Using BLE UI Classes

```
#import <CoreAudioKit/CoreAudioKit.h>
```

Searching for and connecting to external devices

```
CABTMIDICentralViewController *viewController =
    [[CABTMIDICentralViewController alloc] init];
[self.navigationController pushViewController: viewController animated: YES];
```

Using BLE UI Classes

```
#import <CoreAudioKit/CoreAudioKit.h>
```

Searching for and connecting to external devices

```
CABTMIDICentralViewController *viewController =
    [[CABTMIDICentralViewController alloc] init];
[self.navigationController pushViewController: viewController animated: YES];
```

Inter-App Audio

User interface elements

Inter-App Audio

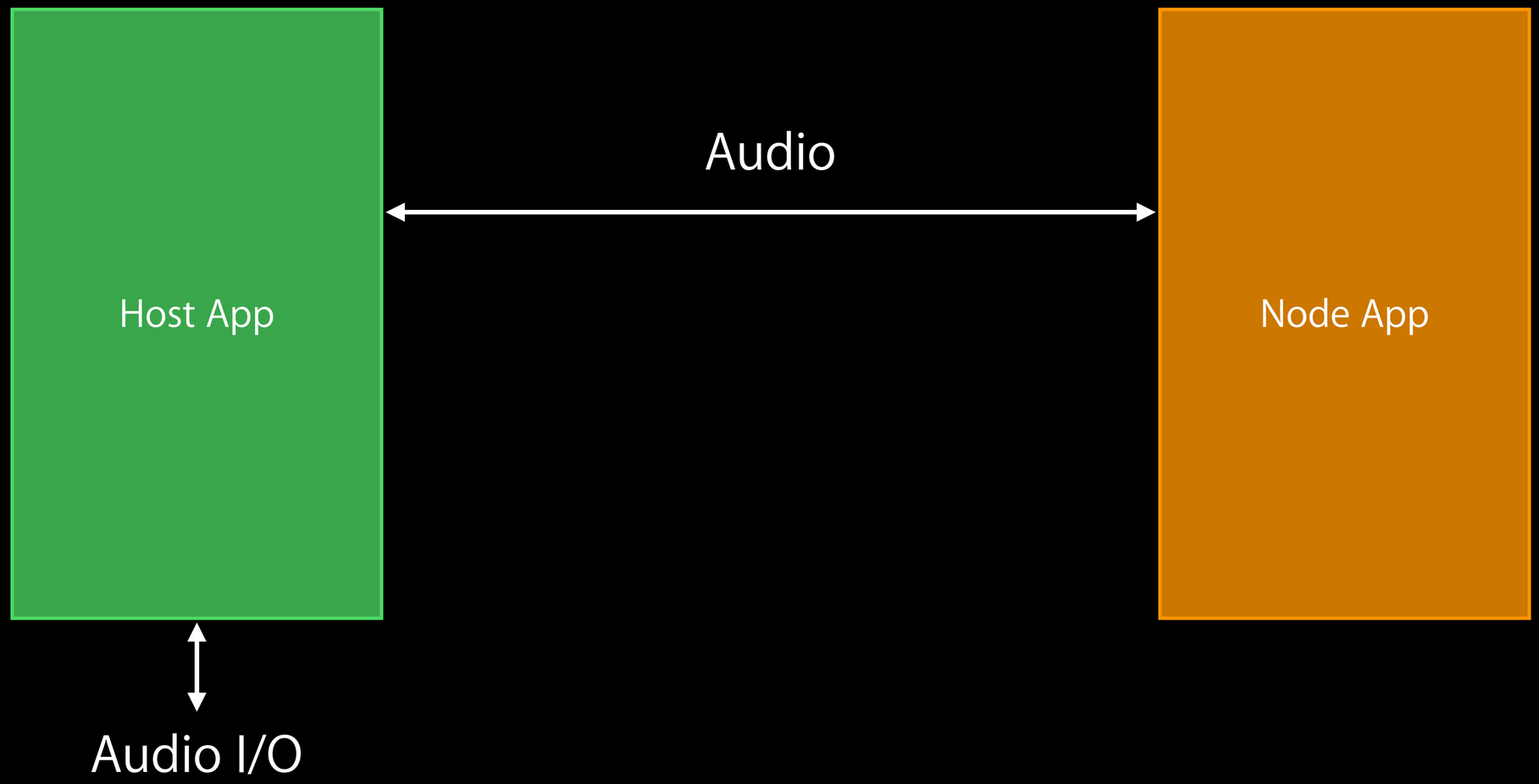
Review

Streams audio between apps in realtime on iOS

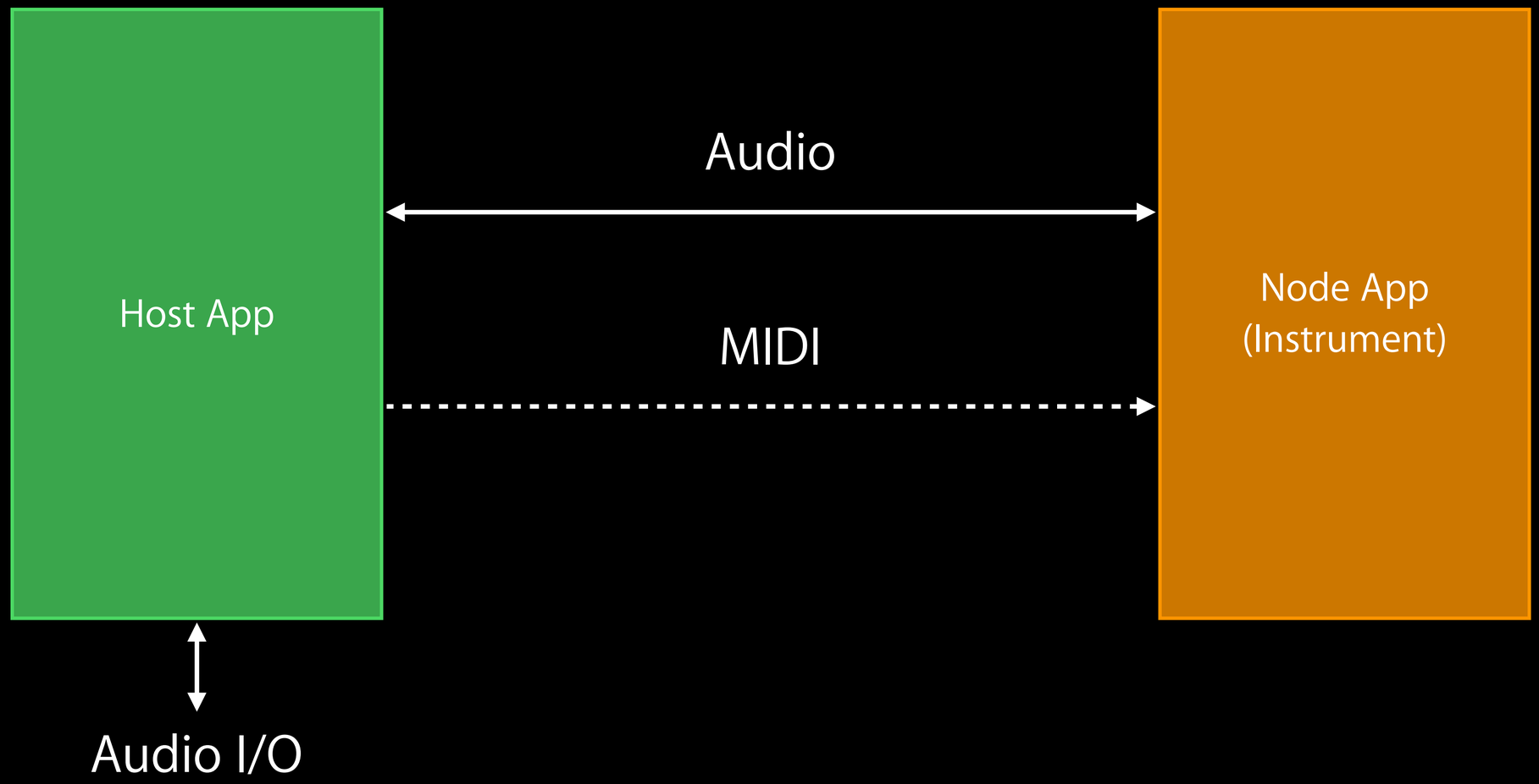
Node apps can be discovered by host apps

See WWDC 2013 Session 206—What's New in Core Audio for iOS

Inter-App Audio



Inter-App Audio



Inter-App Audio User Interface Elements

Overview

Inter-App Audio Switcher

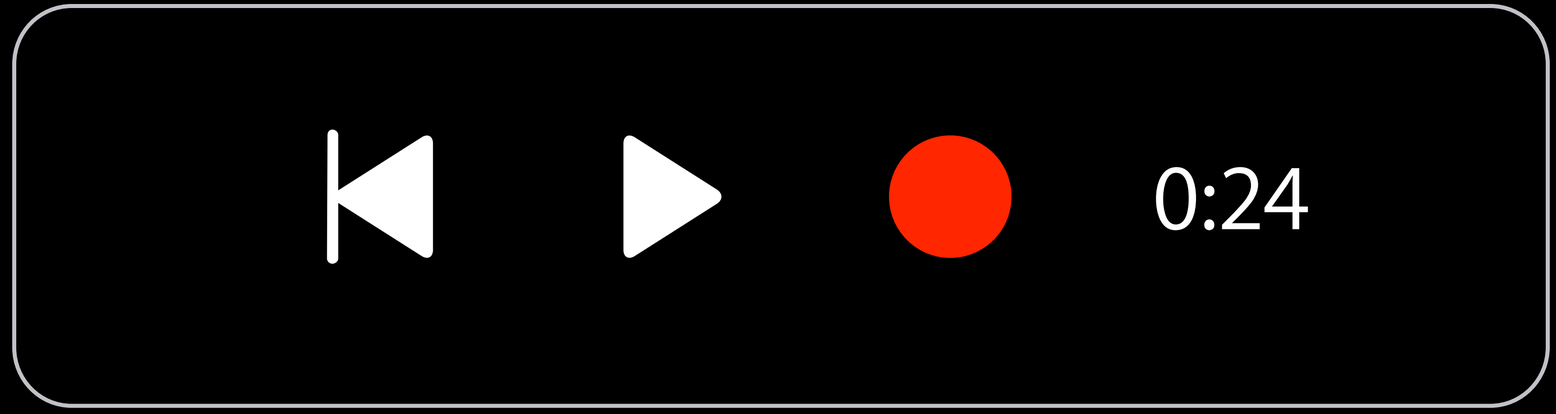
- Provides a simple way to switch between connected apps

Inter-App Audio User Interface Elements

Overview

Inter-App Audio Host Transport

- Displays host transport (play/pause, rewind, and record) controls



Demo

Inter-App Audio user interface elements

Michael Hopkins
Core Audio UI Maestro

Inter-App Audio User Interface Elements

Provides a consistent user experience

Flexible sizing

Minimal code required to adopt

Subclass of UIView

CAInterAppAudioSwitcherView

Creating from a nib file

```
#import <CoreAudioKit/CoreAudioKit.h>

IBOutlet CAInterAppAudioSwitcherView *switcherView;

-(void) viewDidLoad {
    [super viewDidLoad];
    …
    switcherView.backgroundColor = [UIColor darkGrayColor];
    [switcherView setOutputAudioUnit: audioUnit];
}
```

CAInterAppAudioSwitcherView

Creating from a nib file

```
#import <CoreAudioKit/CoreAudioKit.h>

IBOutlet CAInterAppAudioSwitcherView *switcherView;

-(void) viewDidLoad {
    [super viewDidLoad];
    …
    switcherView.backgroundColor = [UIColor darkGrayColor];
    [switcherView setOutputAudioUnit: audioUnit];
}
```

CAInterAppAudioSwitcherView

Creating from a nib file

```
#import <CoreAudioKit/CoreAudioKit.h>

IBOutlet CAInterAppAudioSwitcherView *switcherView;

-(void) viewDidLoad {
    [super viewDidLoad];
    …
    switcherView.backgroundColor = [UIColor darkGrayColor];
    [switcherView setOutputAudioUnit: audioUnit];
}
```

CAInterAppAudioSwitcherView

Creating from a nib file

```
#import <CoreAudioKit/CoreAudioKit.h>

IBOutlet CAInterAppAudioSwitcherView *switcherView;

-(void) viewDidLoad {
    [super viewDidLoad];
    …
    switcherView.backgroundColor = [UIColor darkGrayColor];
    [switcherView setOutputAudioUnit: outputAU];
}
```

CAInterAppAudioTransportView

Creating programmatically

```
CAInterAppAudioTransportView *transportView =
    [[CAInterAppAudioTransportView alloc] initWithFrame:
       CGRectMake(0, 0, kTransportViewHeight, kTransportViewWidth)];

transportView.rewindButtonColor = transportView.playButtonColor =
   transportView.pauseButtonColor = [UIColor whiteColor];
transportView.labelColor = [UIColor lightGrayColor];
transportView.backgroundColor = [UIColor darkGrayColor];
[transportView setOutputAudioUnit: outputAU];

[self addSubview: transportView];
```

CAInterAppAudioTransportView

Creating programmatically

```
CAInterAppAudioTransportView *transportView =
    [[CAInterAppAudioTransportView alloc] initWithFrame:
       CGRectMake(0, 0, kTransportViewHeight, kTransportViewWidth)];

transportView.rewindButtonColor = transportView.playButtonColor =
   transportView.pauseButtonColor = [UIColor whiteColor];
transportView.labelColor = [UIColor lightGrayColor];
transportView.backgroundColor = [UIColor darkGrayColor];
[transportView setOutputAudioUnit: outputAU];

[self addSubview: transportView];
```

CAInterAppAudioTransportView

Creating programmatically

```
CAInterAppAudioTransportView *transportView =
    [[CAInterAppAudioTransportView alloc] initWithFrame:
        CGRectMake(0, 0, kTransportViewHeight, kTransportViewWidth)];

transportView.rewindButtonColor = transportView.playButtonColor =
    transportView.pauseButtonColor = [UIColor whiteColor];
transportView.labelColor = [UIColor lightGrayColor];
transportView.backgroundColor = [UIColor darkGrayColor];
[transportView setOutputAudioUnit: outputAU];

[self addSubview: transportView];
```

CAInterAppAudioTransportView

Creating programmatically

```
CAInterAppAudioTransportView *transportView =
    [[CAInterAppAudioTransportView alloc] initWithFrame:
       CGRectMake(0, 0, kTransportViewHeight, kTransportViewWidth)];

transportView.rewindButtonColor = transportView.playButtonColor =
    transportView.pauseButtonColor = [UIColor whiteColor];
transportView.labelColor = [UIColor lightGrayColor];
transportView.backgroundColor = [UIColor darkGrayColor];
[transportView setOutputAudioUnit: outputAU];

[self addSubview: transportView];
```

CAInterAppAudioTransportView

Creating programmatically

```
CAInterAppAudioTransportView *transportView =
    [[CAInterAppAudioTransportView alloc] initWithFrame:
       CGRectMake(0, 0, kTransportViewHeight, kTransportViewWidth)];

transportView.rewindButtonColor = transportView.playButtonColor =
    transportView.pauseButtonColor = [UIColor whiteColor];
transportView.labelColor = [UIColor lightGrayColor];
transportView.backgroundColor = [UIColor darkGrayColor];
[transportView setOutputAudioUnit: outputAU];

[self addSubview: transportView];
```

Managing Audio Units

Using AVAudioUnitComponentManager

Michael Hopkins
Audio Unit Wrangler

AVAudioUnitComponentManager

Introduction

Objective-C based API for Audio Unit host applications

Querying methods for Audio Units

Simple API for getting information about Audio Units

Audio Unit tagging facilities

Centralized Audio Unit cache

AVAudioUnitComponentManager API

New classes in AV Foundation

AVAudioUnitComponentManager

- Provides multiple search mechanisms
  - NSPredicates
  - Block-based
  - Backwards-compatibility mode using AudioComponentDescriptions

AVAudioUnitComponent

- Provides information about individual Audio Units

Getting Information About an Audio Unit

Finding all stereo effects using AudioComponent API

```
#include <AudioUnit/AUComponent.h>
#include <AudioUnit/AudioUnit.h>
#include <AudioUnit/AudioUnitProperties.h>

AudioComponentDescription desc = {kAudioUnitType_Effect, 0, 0, 0, 0};
AudioComponent comp = NULL;
while (comp = AudioComponentFindNext(comp, &desc)) {
    AudioUnit au;
    if (AudioComponentInstanceNew(comp, &au) != noErr) {
        AudioStreamBasicDescription inputDesc, outputDesc;
        UInt32 dataSize = sizeof(AudioStreamBasicDescription);
        OSStatus result1 = AudioUnitGetProperty(au,
            kAudioUnitProperty_StreamFormat,
            kAudioUnitScope_Output, 0, &inputDesc, &dataSize);
```

Getting Information About an Audio Unit

Finding all stereo effects continued

```
OSStatus result2 = AudioUnitGetProperty(au,
    kAudioUnitProperty_StreamFormat,
    kAudioUnitScope_Input, 0, &outputDesc, &dataSize);
if (result1 != noErr || result2 != noErr)
    continue;
        if (inputDesc.mChannelsPerFrame == 2 &&
            outputDesc.mChannelsPerFrame == 2) {
            // do something with the component to store it into a list
            // then close the Audio Unit
        }
    }
}
```

Getting Information About an Audio Unit

Finding all stereo effects using AVAudioUnitComponentManager API

```
#import <AVFoundation/AVAudioUnitComponent.h>

NSArray *components = [[AVAudioUnitComponentManager sharedAudioUnitManager]
    componentsPassingTest: ^(AVAudioUnitComponent *comp, BOOL *stop) {
        return [[comp typeName] isEqualToString: AVAudioUnitTypeEffect] &&
               [comp supportsNumberInputChannels: 2 outputChannels: 2];
    }];
```

Getting Information About an Audio Unit

Finding all stereo effects using AVAudioUnitComponentManager API

```
#import <AVFoundation/AVAudioUnitComponent.h>

NSArray *components = [[AVAudioUnitComponentManager sharedAudioUnitManager]
    componentsPassingTest: ^(AVAudioUnitComponent *comp, BOOL *stop) {
        return [[comp typeName] isEqualToString: AVAudioUnitTypeEffect] &&
               [comp supportsNumberInputChannels: 2 outputChannels: 2];
    }];
```

Getting Information About an Audio Unit

Finding all stereo effects using AVAudioUnitComponentManager API

```
#import <AVFoundation/AVAudioUnitComponent.h>

NSArray *components = [[AVAudioUnitComponentManager sharedAudioUnitManager]
    componentsPassingTest: ^(AVAudioUnitComponent *comp, BOOL *stop) {
        return [[comp typeName] isEqualToString: AVAudioUnitTypeEffect] &&
               [comp supportsNumberInputChannels: 2 outputChannels: 2];
    }];
```

Getting Information About an Audio Unit

Finding all stereo effects using AVAudioUnitComponentManager API

```
#import <AVFoundation/AVAudioUnitComponent.h>

NSArray *components = [[AVAudioUnitComponentManager sharedAudioUnitManager]
    componentsPassingTest: ^(AVAudioUnitComponent *comp, BOOL *stop) {
        return [[comp typeName] isEqualToString: AVAudioUnitTypeEffect] &&
               [comp supportsNumberInputChannels: 2 outputChannels: 2];
    }];
```

Getting Information About an Audio Unit

Finding all stereo effects using AVAudioUnitComponentManager API

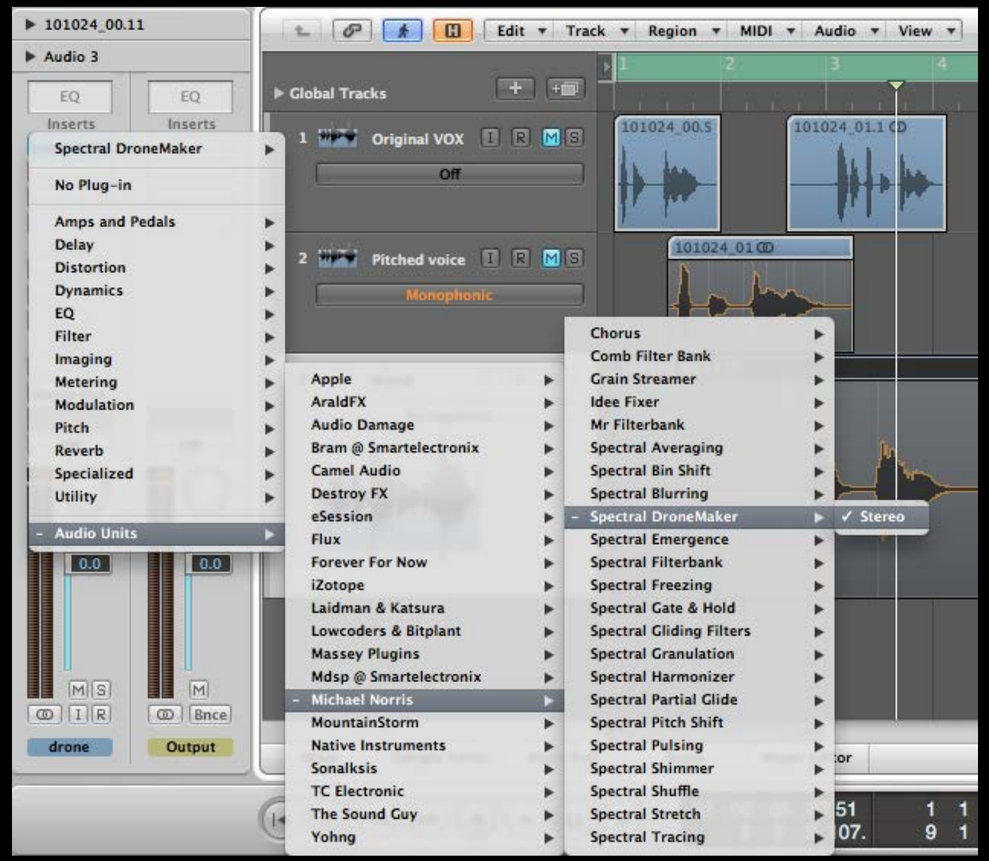
```
#import <AVFoundation/AVAudioUnitComponent.h>

NSArray *components = [[AVAudioUnitComponentManager sharedAudioUnitManager]
    componentsPassingTest: ^(AVAudioUnitComponent *comp, BOOL *stop) {
        return [[comp typeName] isEqualToString: AVAudioUnitTypeEffect] &&
               [comp supportsNumberInputChannels: 2 outputChannels: 2];
}];
```

Tagging Audio Units

Introduction

Many users have lots of Audio Units…

Finding the right one can be difficult



AVAudioUnitComponentManager

Tags

Audio Units can have one or more tags

System tags

- Defined by creator of Audio Unit (read-only)

User tags

- Specified by current user (read/write)
- Each user can have a unique set of tags

AVAudioUnitComponentManager

Tags continued

A tag is a localized string

- Arbitrary
- Predefined (AudioComponent.h)
  - Type (equalizer, dynamics, distortion, etc.)
  - Usage (drums, guitar, vocal, etc.)

Demo

Audio Unit tags in AU Lab

Michael Hopkins
Audio Unit Scientist

Finding Audio Units with a Specific Tag

```
#import <AVFoundation/AVAudioUnitComponent.h>

NSString *aString = @“My Favorite Tag”;
NSPredicate *aPredicate = [NSPredicate predicateWithFormat:
      @"allTagNames CONTAINS[cd] %@", aString];

NSArray *components = [[AVAudioUnitComponentManager sharedAudioUnitManager]
    componentsMatchingPredicate: aPredicate];
```

Finding Audio Units with a Specific Tag

```
#import <AVFoundation/AVAudioUnitComponent.h>

NSString *aString = @“My Favorite Tag”;
NSPredicate *aPredicate = [NSPredicate predicateWithFormat:
      @"allTagNames CONTAINS[cd] %@", aString];

NSArray *components = [[AVAudioUnitComponentManager sharedAudioUnitManager]
    componentsMatchingPredicate: aPredicate];
```

Finding Audio Units with a Specific Tag

```
#import <AVFoundation/AVAudioUnitComponent.h>

NSString *aString = @“My Favorite Tag”;
NSPredicate *aPredicate = [NSPredicate predicateWithFormat:
        @"allTagNames CONTAINS[cd] %@", aString];

NSArray *components = [[AVAudioUnitComponentManager sharedAudioUnitManager]
    componentsMatchingPredicate: aPredicate];
```

Finding Audio Units with a Specific Tag

```
#import <AVFoundation/AVAudioUnitComponent.h>

NSString *aString = @“My Favorite Tag”;
NSPredicate *aPredicate = [NSPredicate predicateWithFormat:
      @"allTagNames CONTAINS[cd] %@", aString];

NSArray *components = [[AVAudioUnitComponentManager sharedAudioUnitManager]
    componentsMatchingPredicate: aPredicate];
```

Finding Audio Units with a Specific Tag

```
#import <AVFoundation/AVAudioUnitComponent.h>

NSString *aString = @“My Favorite Tag”;
NSPredicate *aPredicate = [NSPredicate predicateWithFormat:
      @"allTagNames CONTAINS[cd] %@", aString];

NSArray *components = [[AVAudioUnitComponentManager sharedAudioUnitManager]
    componentsMatchingPredicate: aPredicate];
```

Tagging Facilities

Getting user tags of an AVAudioUnitComponent

```
NSArray *userTags = aComp.userTagNames;
```

Tagging Facilities

Getting user tags of an AVAudioUnitComponent

```
NSArray *userTags = aComp.userTagNames;
```

Tagging Facilities

Getting user tags of an AVAudioUnitComponent

```
NSArray *userTags = aComp.userTagNames;
```

Setting user tags of an AVAudioUnitComponent

```
aComp.userTagNames = @[@“Trippy”,  @“Favorites”];
```

Tagging Facilities

Getting user tags of an AVAudioUnitComponent

```
NSArray *userTags = aComp.userTagNames;
```

Setting user tags of an AVAudioUnitComponent

```
aComp.userTagNames = @[@“Trippy”,  @“Favorites”];
```

Tagging Facilities

Getting user tags of an AVAudioUnitComponent

```
NSArray *userTags = aComp.userTagNames;
```

Setting user tags of an AVAudioUnitComponent

```
aComp.userTagNames = @[@“Trippy”,  @“Favorites”];
```

Getting all tags of an AVAudioUnitComponent

```
NSArray *allTags = aComp.allTagNames;
```

Tagging Facilities

Getting user tags of an AVAudioUnitComponent

```
NSArray *userTags = aComp.userTagNames;
```

Setting user tags of an AVAudioUnitComponent

```
aComp.userTagNames = @[@“Trippy”,  @“Favorites”];
```

Getting all tags of an AVAudioUnitComponent

```
NSArray *allTags = aComp.allTagNames;
```

More Tagging Facilities

Getting a localized list of standard system tags

```
AVAudioUnitComponentManager *manager = [AVAudioUnitComponentManager
sharedAudioUnitComponentManager];

NSArray *standardTags = manager.standardLocalizedTagNames;
```

More Tagging Facilities

Getting a localized list of standard system tags

```
AVAudioUnitComponentManager *manager = [AVAudioUnitComponentManager
sharedAudioUnitComponentManager];

NSArray *standardTags = manager.standardLocalizedTagNames;
```

More Tagging Facilities

Getting a localized list of standard system tags

```
AVAudioUnitComponentManager *manager = [AVAudioUnitComponentManager
sharedAudioUnitComponentManager];

NSArray *standardTags = manager.standardLocalizedTagNames;
```

More Tagging Facilities

Getting a localized list of standard system tags

```
AVAudioUnitComponentManager *manager = [AVAudioUnitComponentManager
sharedAudioUnitComponentManager];

NSArray *standardTags = manager.standardLocalizedTagNames;
```

Getting a complete list of all available localized tags

```
NSArray *allTags = manager.tagNames;
```

More Tagging Facilities

Getting a localized list of standard system tags

```
AVAudioUnitComponentManager *manager = [AVAudioUnitComponentManager
sharedAudioUnitComponentManager];

NSArray *standardTags = manager.standardLocalizedTagNames;
```

Getting a complete list of all available localized tags

```
NSArray *allTags = manager.tagNames;
```

Adding Built-in Tags to an AudioUnit

Add a tags section to Info.plist of AudioComponent bundle

```
<key>tags</key>
<array>
    <string>Effect</string>
    <string>Equalizer</string>
    <string>Custom Tag</string>
</array>
```

Adding Built-in Tags to an AudioUnit

Add a tags section to Info.plist of AudioComponent bundle

```
<key>tags</key>
<array>
    <string>Effect</string>
    <string>Equalizer</string>
    <string>Custom Tag</string>
</array>
```

Adding Built-in Tags to an AudioUnit

Add a tags section to Info.plist of AudioComponent bundle

```
<key>tags</key>
<array>
    <string>Effect</string>
    <string>Equalizer</string>
    <string>Custom Tag</string>
</array>
```

Adding Built-in Tags to an AudioUnit

Add a tags section to Info.plist of AudioComponent bundle

```
<key>tags</key>
<array>
    <string>Effect</string>
    <string>Equalizer</string>
    <string>Custom Tag</string>
</array>
```

Localize Custom Tag by adding AudioUnitTags.strings file in bundle

```
“Custom Tag” = “Localized Tag”;
```

Adding Built-in Tags to an AudioUnit

Add a tags section to Info.plist of AudioComponent bundle

```
<key>tags</key>
<array>
    <string>Effect</string>
    <string>Equalizer</string>
    <string>Custom Tag</string>
</array>
```

Localize Custom Tag by adding AudioUnitTags.strings file in bundle

```
“Custom Tag” = “Localized Tag”;
```

Adding Built-in Tags to an AudioUnit

Add a tags section to Info.plist of AudioComponent bundle

```
<key>tags</key>
<array>
    <string>Effect</string>
    <string>Equalizer</string>
    <string>Custom Tag</string>
</array>
```

Localize Custom Tag by adding AudioUnitTags.strings file in bundle

```
“Custom Tag” = “Localized Tag”;
```

Do not localize Standard System Tags!

Action Items

Users

- Tag your Audio Units

Host developers

- Adopt AVAudioUnitComponentManager API
- Surprise and delight your users

Audio Unit developers

- Add system tags to your Audio Units

AVAudioSession Tips

Eric Johnson
Audio Traffic Controller

References

Updated audio session programming guide

https://developer.apple.com/library/ios/documentation/Audio/Conceptual/AudioSessionProgrammingGuide

WWDC 2012 Session 505—Audio Session and Multiroute Audio in iOS

Managing Session Activation State

App state vs. audio session activation state

App state

- Not running
- Foreground inactive/foreground active
- Background
- Suspended

Audio session activation state

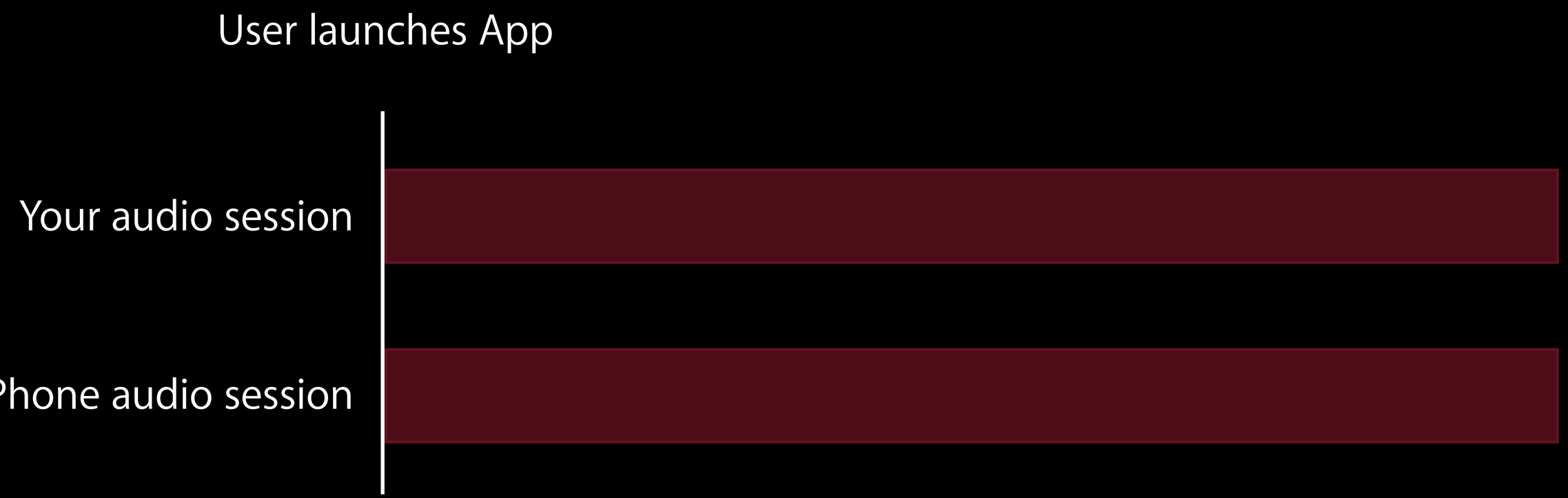
- Active or inactive
- Interruptions

Session State Changes

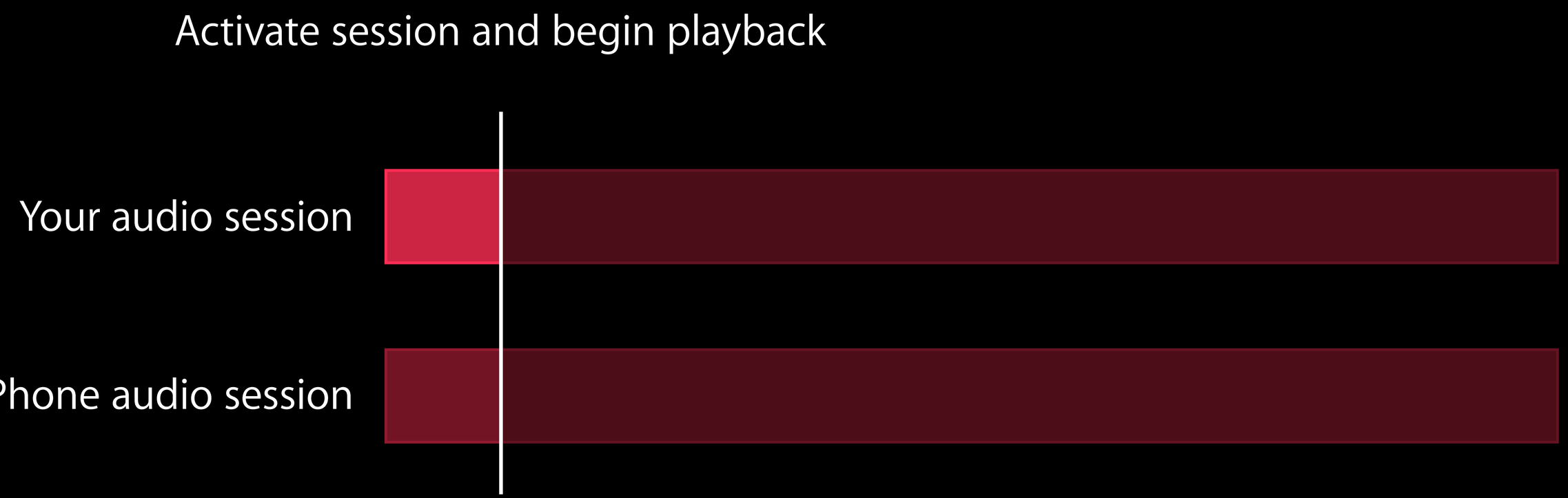
Your audio session

Phone audio session

Session State Changes



Session State Changes

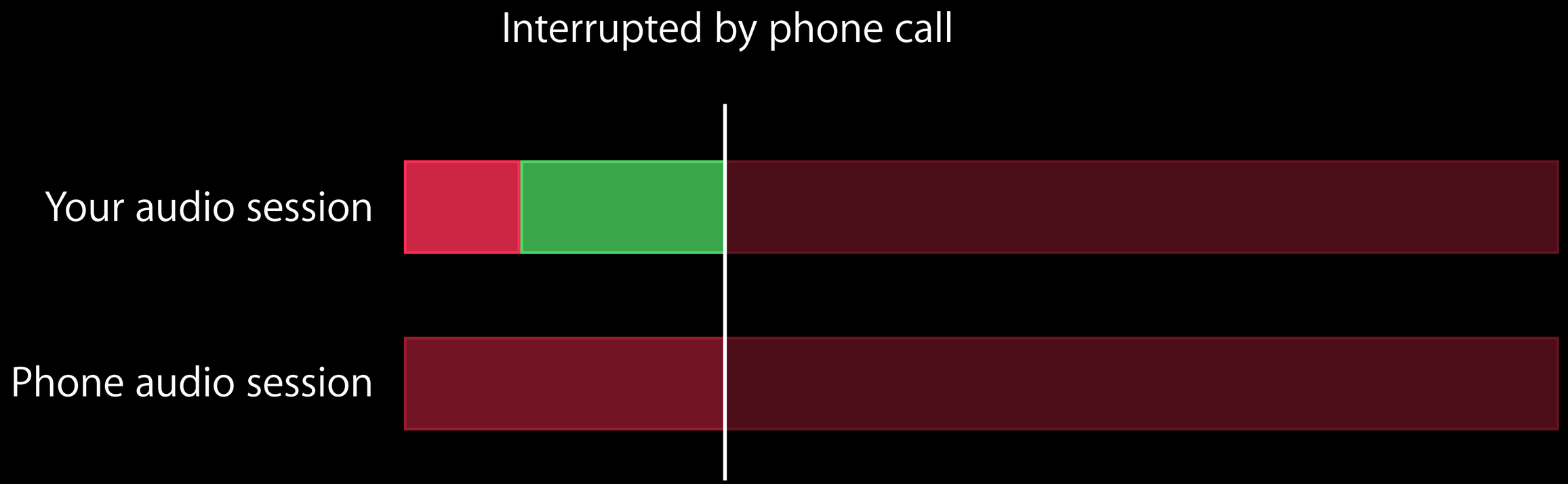


Session State Changes

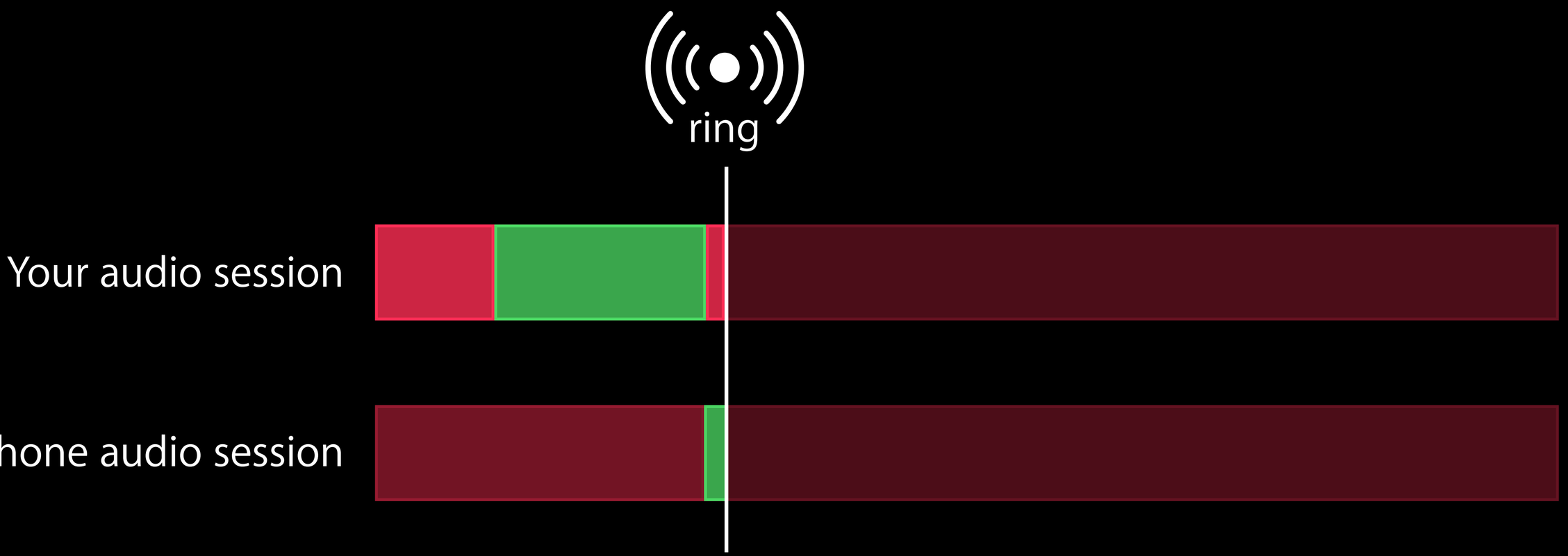
Your audio session

Phone audio session

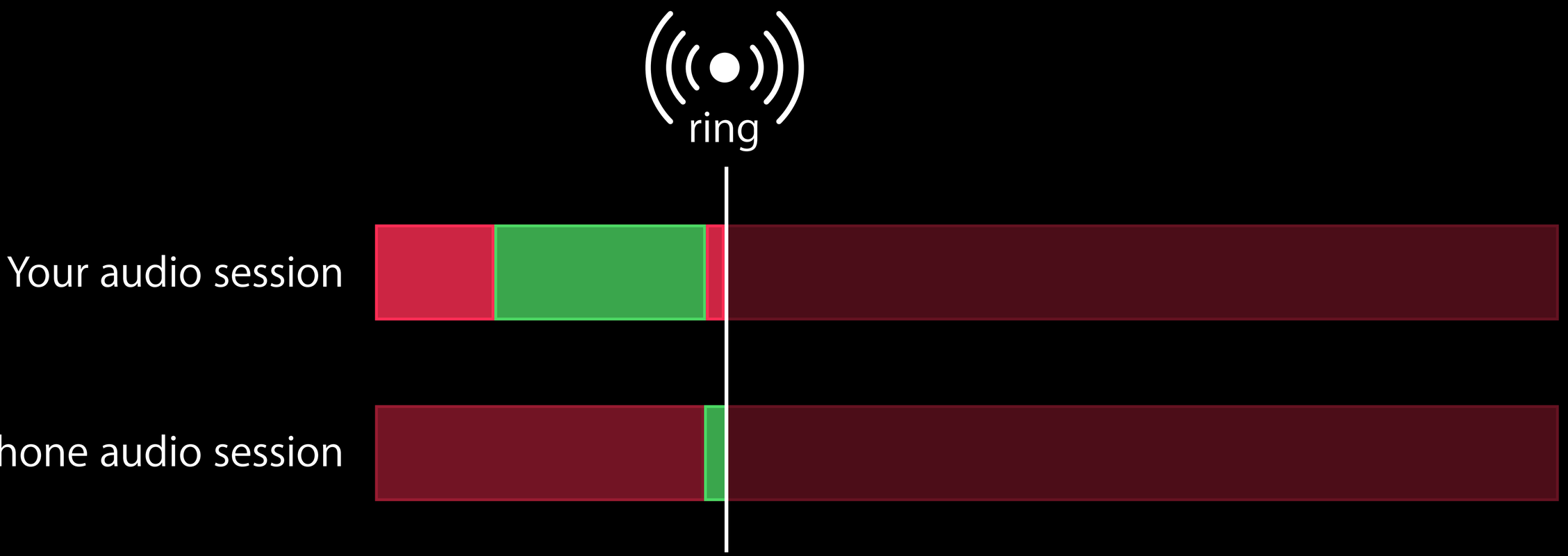
Session State Changes



Session State Changes



Session State Changes

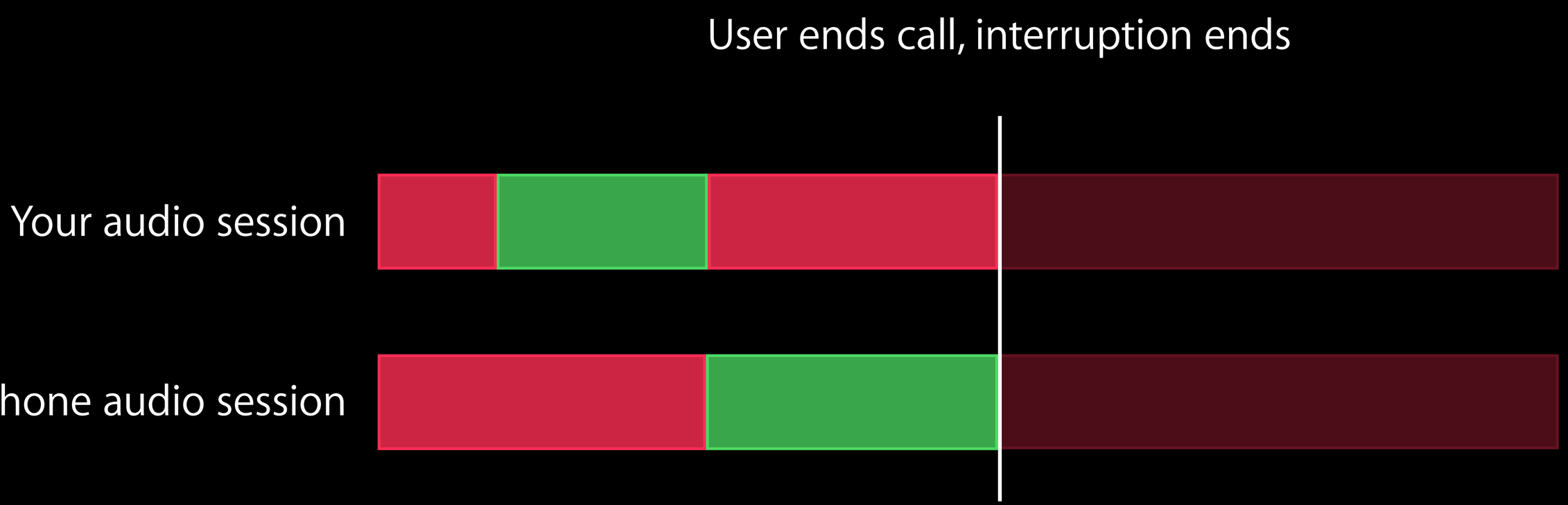


Session State Changes

Your audio session

Phone audio session

Session State Changes



Session State Changes

Reactivate session, resume playback

Your audio session

Phone audio session

Managing Session Activation State

Needs vary by application type

Games

Media playback

VoIP and chat apps

Metering and monitoring apps

Browser-like apps

Navigation and fitness apps

Music-making apps

Managing Session Activation State

Activation

Game apps

- Go active upon app delegate's applicationDidBecomeActive: method
- Handle interruptions
  - Begin—Update internal state
  - End—Go active and resume audio playback

Managing Session Activation State

Activation

Media playback apps

- Music, podcasts, streaming radio
- Go active when the user presses “play” button
- Stay active, unless interrupted
- Handle interruptions
  - Begin—Update UI and internal state
  - End—Honor AVAudioSessionInterruptionOptionShouldResume

Managing Session Activation State

Deactivation

End ducking of other audio

- Navigation/Fitness

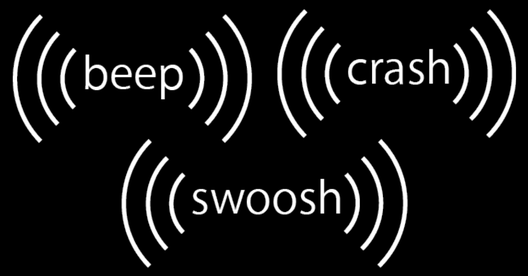
Allow other audio to resume

- VoIP/Chat
- Short videos in browser views
- AVAudioSessionSetActiveOption**NotifyOthersOnDeactivation**

Muting Secondary Audio

Primary vs. secondary audio

Game App



Primary Audio

Secondary Audio

Muting Secondary Audio

secondaryAudioShouldBeSilencedHint

- Hint to foreground app for silencing secondary audio
- Check in app delegate's applicationDidBecomeActive: method

Muting Secondary Audio

secondaryAudioShouldBeSilencedHint

- Hint to foreground app for silencing secondary audio
- Check in app delegate's applicationDidBecomeActive: method

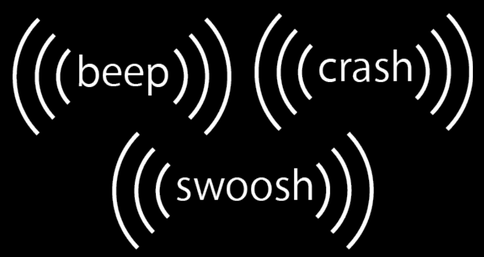
AVAudioSessionSilenceSecondaryAudioHintNotification

- Delivered to foreground, active audio sessions
- Begin—Mute secondary audio
- End—Resume secondary audio

Muting Secondary Audio

Game App
(Foreground)

Music App
(Background)

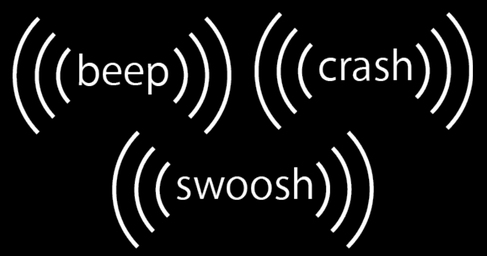


Muting Secondary Audio

Game App
(Foreground)

Music App
(Background)

Begin Playing



Muting Secondary Audio

Game App
(Foreground)

Notification

iOS

Begin Playing

Music App
(Background)

((beep))

((crash))

((swoosh))

Muting Secondary Audio

Game App
(Foreground)

Notification

iOS

Begin Playing

Music App
(Background)

((beep))

((crash))

((swoosh))

Muting Secondary Audio

Game App
(Foreground)

((beep)) ((crash))
((swoosh))

Music App
(Background)

Muting Secondary Audio

Game App
(Foreground)

Music App
(Background)

Pauses

((beep)) ((crash)) ((swoosh))

Muting Secondary Audio

Game App
(Foreground)

Notification

iOS

Pauses

Music App
(Background)

(beep)

(crash)

(swoosh)

Muting Secondary Audio

Game App
(Foreground)

Notification

iOS

Pauses

Music App
(Background)

((beep))

((crash))

((swoosh))

Muting Secondary Audio

Updated best practices

Do not change category based on isOtherAudioPlaying

Use AVAudioSessionCategoryAmbient

Use secondaryAudioShouldBeSilencedHint

Use AVAudioSessionSilenceSecondaryAudioHintNotification

New AV Foundation Audio Classes

Doug Wyatt
Core Audio Syntactic Confectioner

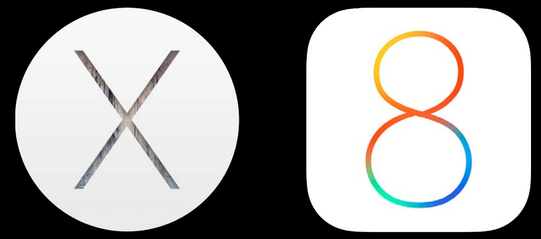
New AV Foundation Audio Classes

Introduction

Tour of the classes

Example

Introduction



Core Audio/AudioToolbox—Powerful but not always simple

Historically—C++ classes in Core Audio SDK

New—Objective-C classes in AVFoundation.framework

Available—OS X 10.10 and iOS 8.0

Goals

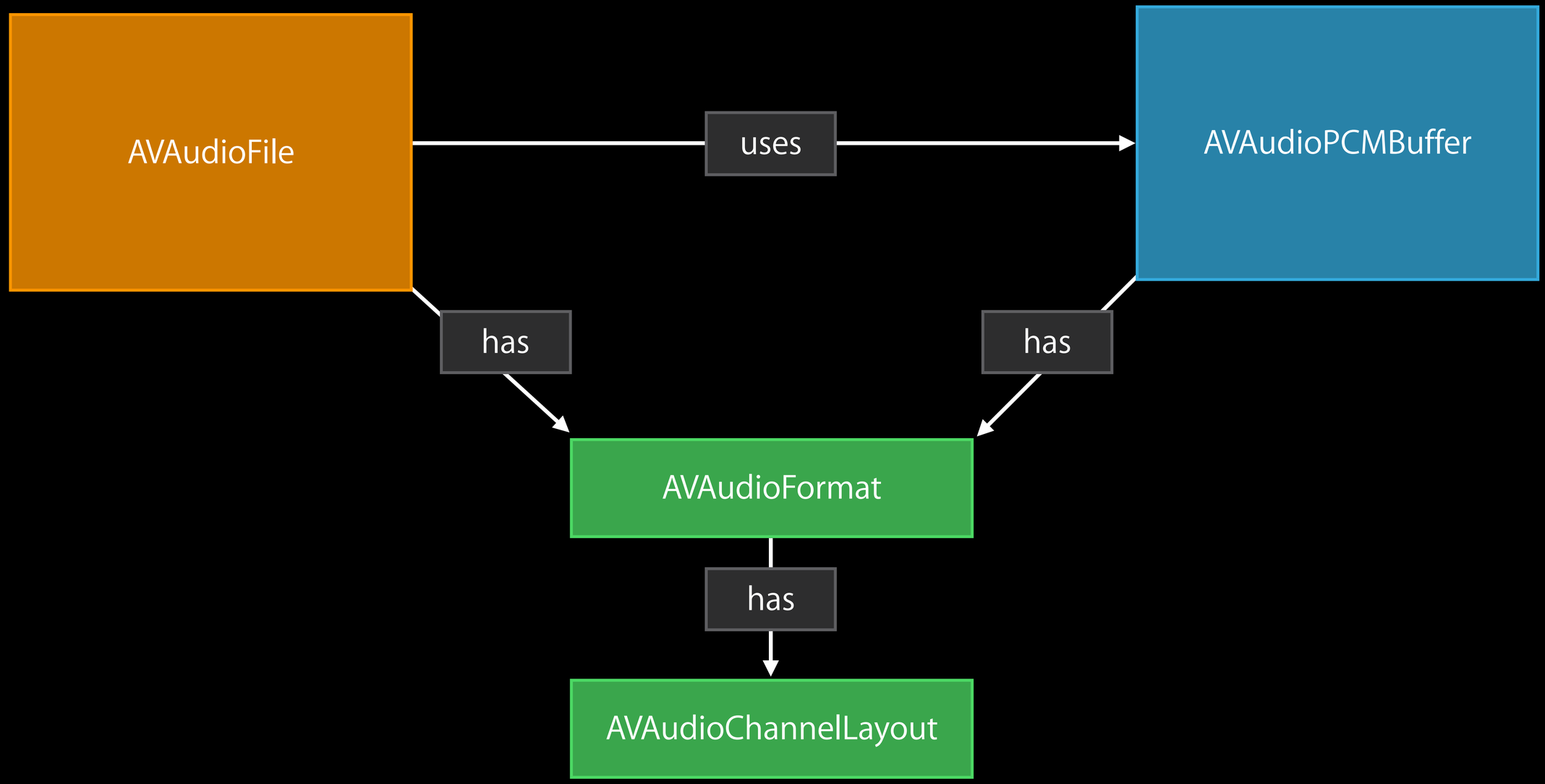
Wrap C structs in Objective-C

- Simpler to build
- Can be passed to low-level APIs

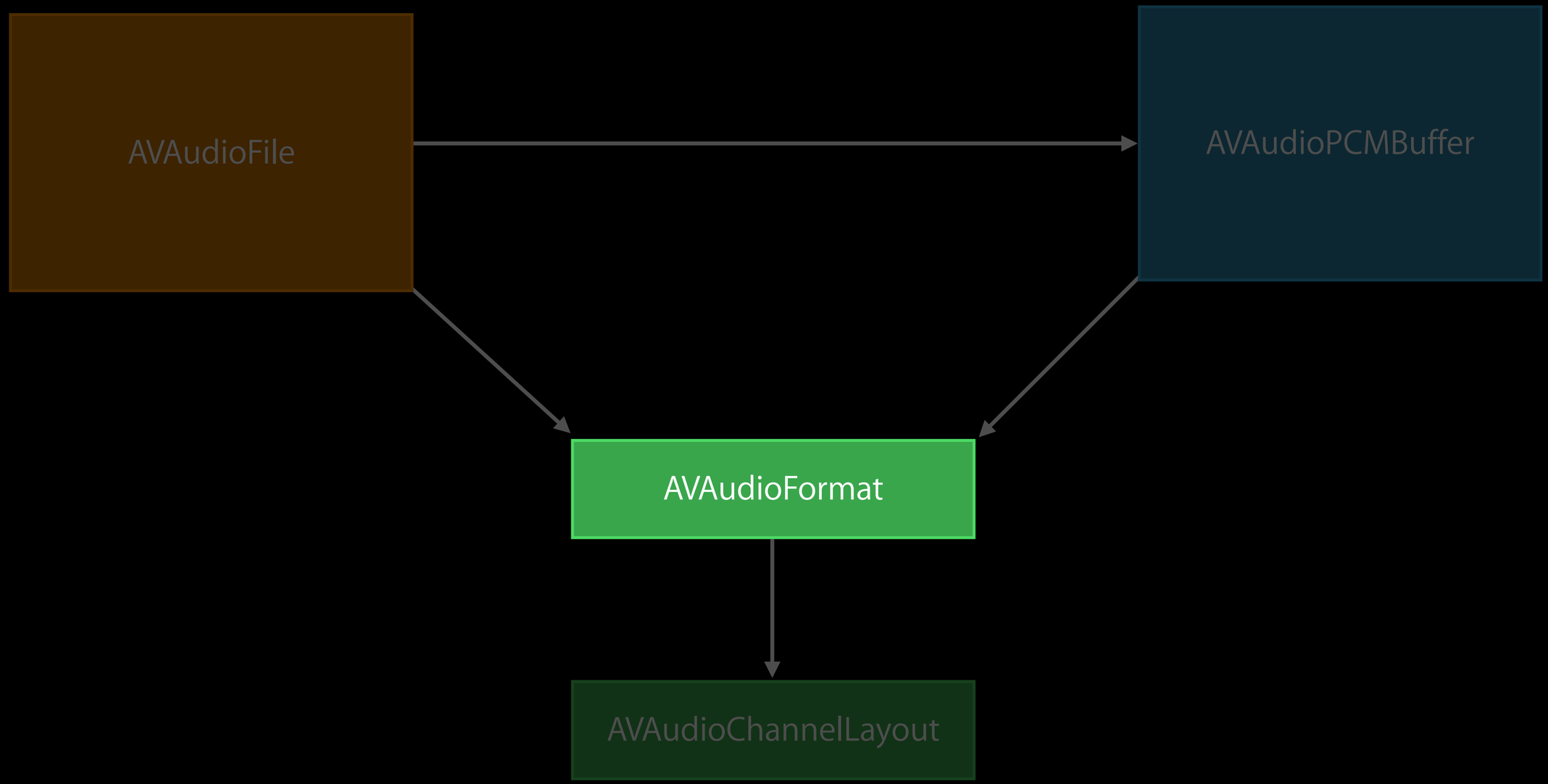
AVAudioEngine

Stay realtime-safe

Classes


AVAudioFile
uses
AVAudioPCMBuffer
has
has
AVAudioFormat
has
AVAudioChannelLayout

AVAudioFormat



AVAudioFormat

Describes data in audio file or stream

Wraps

```
struct AudioStreamBasicDescription {
    Float64 mSampleRate;
    UInt32  mFormatID;
    UInt32  mFormatFlags;
    UInt32  mBytesPerPacket;
    UInt32  mFramesPerPacket;
    UInt32  mBytesPerFrame;
    UInt32  mChannelsPerFrame;
    UInt32  mBitsPerChannel;
    UInt32  mReserved;
};

- (instancetype)initWithStreamDescription:
                    (const AudioStreamBasicDescription *)asbd;

@property (nonatomic, readonly)
                    const AudioStreamBasicDescription *streamDescription;
```

“Standard” Formats

“Canonical”

- Was float on OS X, 8.24 fixed-point on iOS
- Now deprecated

“Standard”

- Non-interleaved 32-bit float
- On both platforms

```
(instancetype)initStandardFormatWithSampleRate:
              (double)sampleRate channels:(AVAudioChannelCount)channels;

@property (nonatomic, readonly, getter=isStandard) BOOL standard;
```

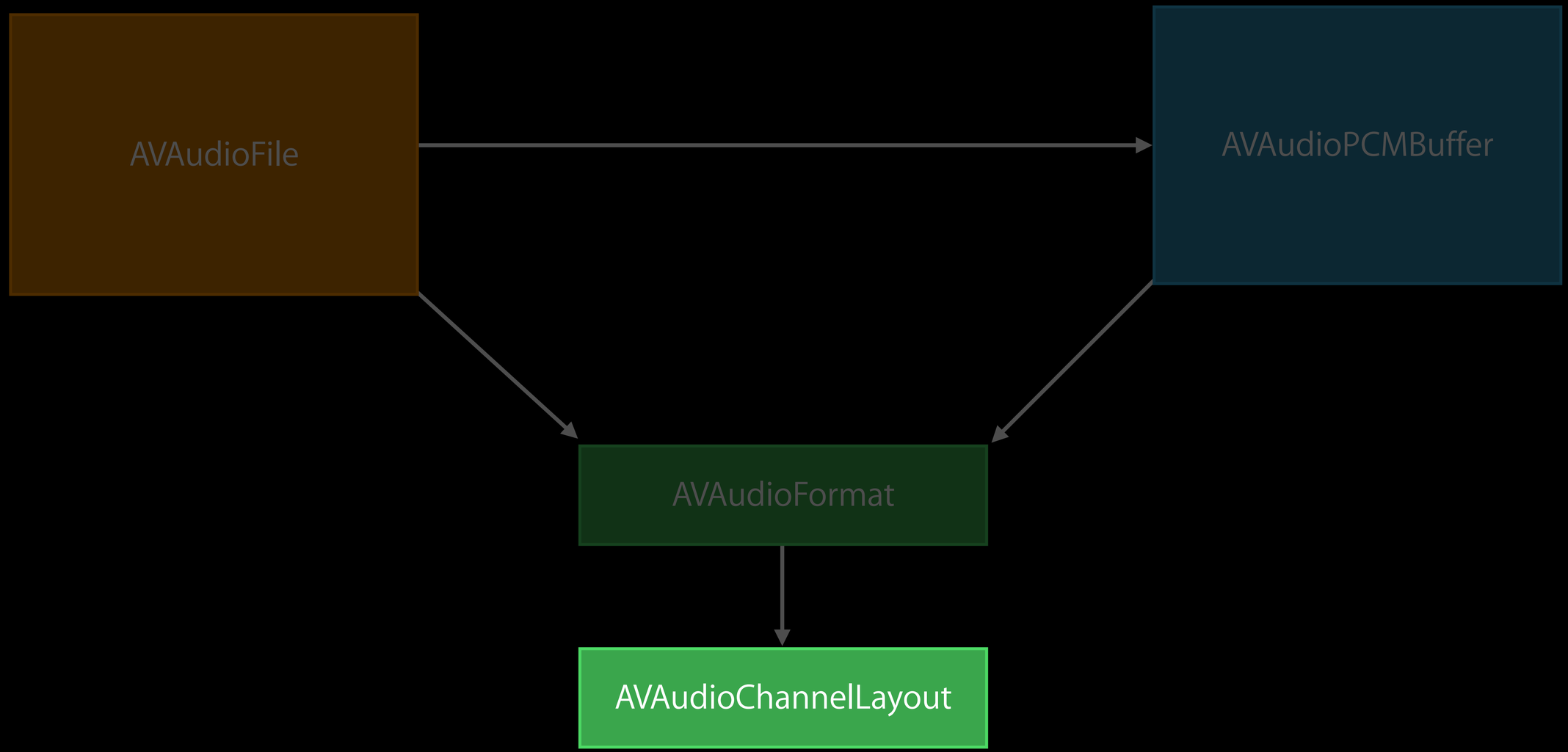
“Common” Formats

Formats often used in signal processing

Always native-endian

```
typedef NS_ENUM(NSUInteger, AVAudioCommonFormat) {
    AVAudioOtherFormat = 0,
    AVAudioPCMFormatFloat32 = 1,
    AVAudioPCMFormatFloat64 = 2,
    AVAudioPCMFormatInt16 = 3,
    AVAudioPCMFormatInt32 = 4
};

- (instancetype)initWithCommonFormat:(AVAudioCommonFormat)format sampleRate:
(double)sampleRate channels:(AVAudioChannelCount)channels interleaved:
(BOOL)interleaved;

@property (nonatomic, readonly) AVAudioCommonFormat commonFormat;
```

AVAudioChannelLayout



AVAudioChannelLayout

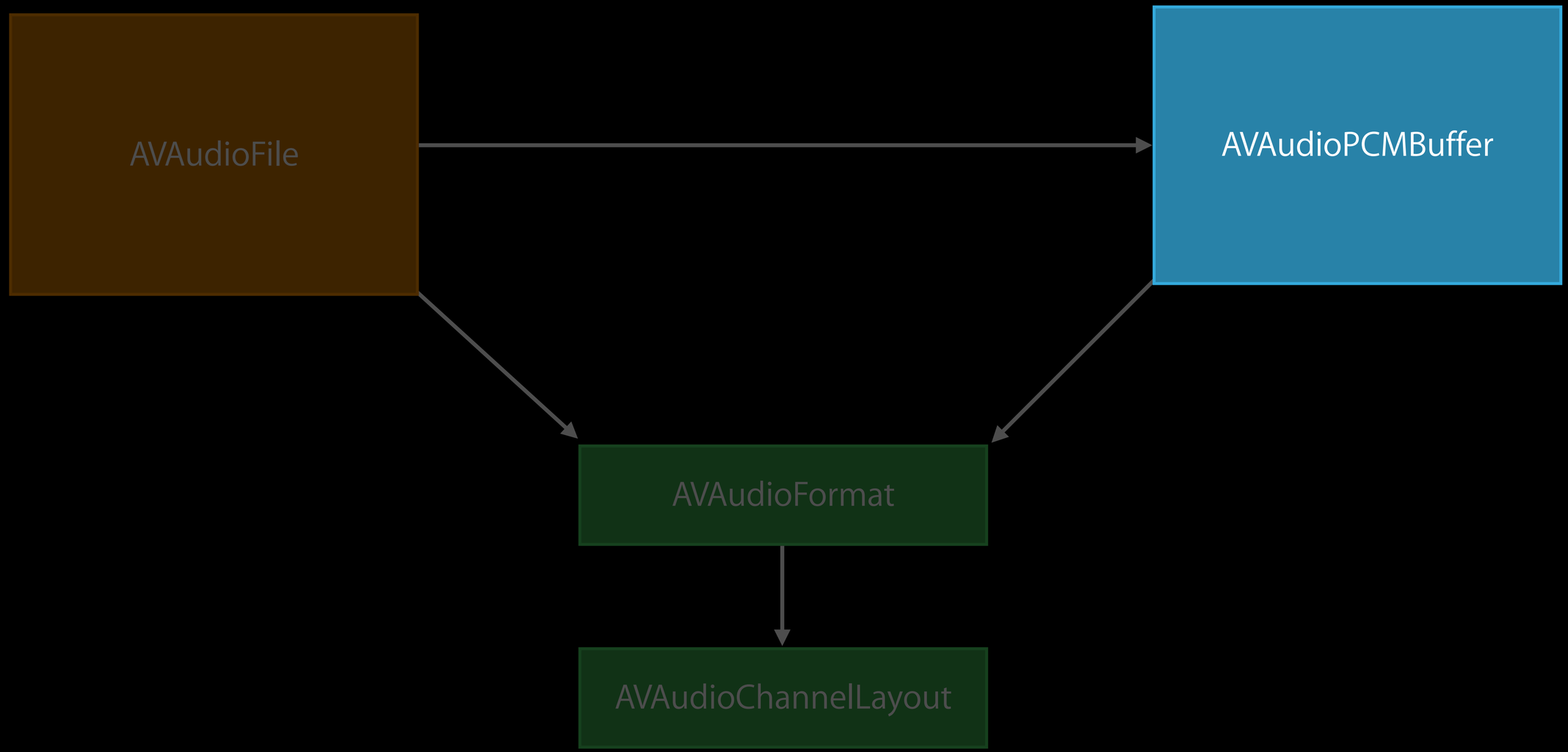
Describes ordering/roles of multiple channels

AVAudioFormat has an AVAudioChannelLayout

AVAudioFormat initializers require a layout when describing > 2 channels

Wraps AudioChannelLayout (CoreAudioTypes.h)

AVAudioPCMBuffer



AVAudioPCMBuffer

Memory for storing audio data in any PCM format

Wraps

```
struct AudioBuffer {
    UInt32  mNumberChannels;
    UInt32  mDataByteSize;
    void *  mData;
};
struct AudioBufferList {
    UInt32      mNumberBuffers;
    AudioBuffer mBuffers[1]; // variable length
};

@property (nonatomic, readonly) const AudioBufferList *audioBufferList;
@property (nonatomic, readonly) AudioBufferList *mutableAudioBufferList;
```

AVAudioPCMBuffer

```
- (instancetype)initWithPCMFormat:(AVAudioFormat *)format frameCapacity:
(AVAudioFrameCount)frameCapacity;
```

Has an AVAudioFormat

```
@property (nonatomic, readonly) AVAudioFormat *format;
```

Has separate frame capacity and length

```
@property (nonatomic, readonly) AVAudioFrameCount frameCapacity;

@property (nonatomic) AVAudioFrameCount frameLength;
```

AVAudioPCMBuffer

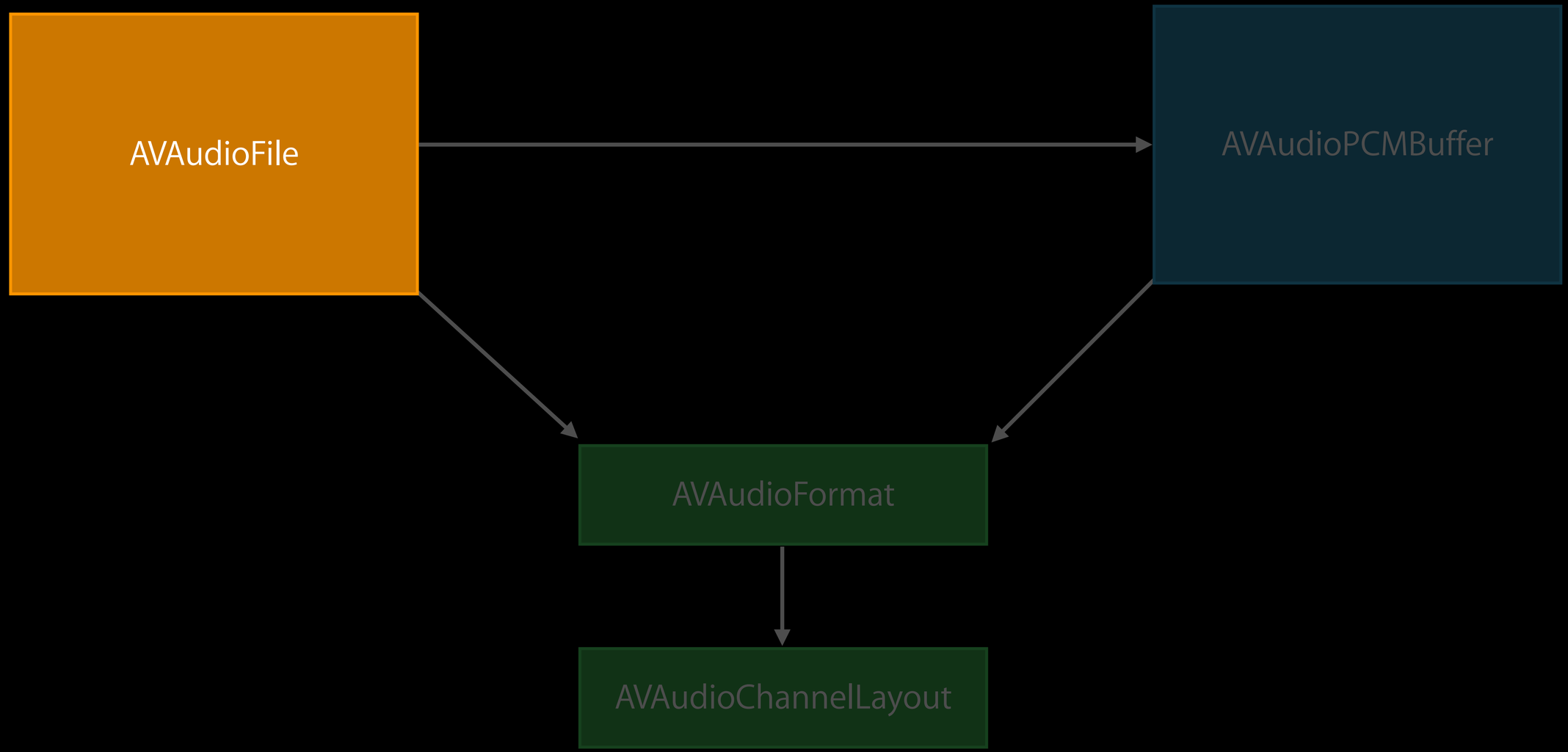
Sample access

```
@property (nonatomic, readonly) float * const *floatChannelData;
@property (nonatomic, readonly) int16_t * const *int16ChannelData;
@property (nonatomic, readonly) int32_t * const *int32ChannelData;
```

Realtime-safety

- Don’t access buffer’s properties on audio I/O thread
- Cache AudioBufferList pointer or one of the above pointer arrays

AVAudioFile



AVAudioFile

Read and write files of any supported format

Transparently decodes while reading, encodes while writing

“Processing format”

- Specified at initialization
- Either standard or common

Similar to ExtAudioFile

AVAudioFile Initializers and Formats

AVAudioFile Initializers and Formats

```
- (instancetype)initForReading:(NSURL *)fileURL error:(NSError **)outError;
```

AVAudioFile Initializers and Formats

```
- (instancetype)initForReading:(NSURL *)fileURL error:(NSError **)outError;

- (instancetype)initForWriting:(NSURL *)fileURL settings:(NSDictionary
*)settings error:(NSError **)outError;
```

AVAudioFile Initializers and Formats

```
- (instancetype)initForReading:(NSURL *)fileURL error:(NSError **)outError;

- (instancetype)initForWriting:(NSURL *)fileURL settings:(NSDictionary
*)settings error:(NSError **)outError;

@property (nonatomic, readonly) AVAudioFormat *fileFormat;
```

AVAudioFile Initializers and Formats

```
- (instancetype)initForReading:(NSURL *)fileURL error:(NSError **)outError;

- (instancetype)initForWriting:(NSURL *)fileURL settings:(NSDictionary
*)settings error:(NSError **)outError;

@property (nonatomic, readonly) AVAudioFormat *fileFormat;
@property (nonatomic, readonly) AVAudioFormat *processingFormat;
```

AVAudioFile I/O

AVAudioFile I/O

```
- (BOOL)readIntoBuffer:(AVAudioPCMBuffer *)buffer error:(NSError **)outError;
```

AVAudioFile I/O

```
- (BOOL)readIntoBuffer:(AVAudioPCMBuffer *)buffer error:(NSError **)outError;

- (BOOL)writeFromBuffer:(const AVAudioPCMBuffer *)buffer error:(NSError
**)outError;
```

AVAudioFile I/O

```
- (BOOL)readIntoBuffer:(AVAudioPCMBuffer *)buffer error:(NSError **)outError;

- (BOOL)writeFromBuffer:(const AVAudioPCMBuffer *)buffer error:(NSError
**)outError;

@property (nonatomic) AVAudioFramePosition framePosition;
```

Example—Reading an Audio File

Open the file

```
static BOOL AudioFileInfo(NSURL *fileURL)
{
    NSError *error = nil;

    AVAudioFile *audioFile = [[AVAudioFile alloc]
    initForReading: fileURL
    commonFormat: AVAudioPCMFormatFloat32
    interleaved: NO
    error: &error];
```

Fetch and print basic info

```
NSLog(@“File URL:          %@\n”, fileURL.absoluteString);
NSLog(@“File format:       %@\n”, audioFile.fileFormat.description);
NSLog(@“Processing format: %@\n”, audioFile.processingFormat.description);

AVAudioFramePosition fileLength = audioFile.length;
 NSLog(@“Length:            %lld frames, %.3f seconds\n", (long
long)fileLength, fileLength / audioFile.fileFormat.sampleRate);
```

Create a buffer to read into

```
 const AVAudioFrameCount kBufferFrameCapacity = 128 * 1024L;
 AVAudioPCMBuffer *readBuffer = [[AVAudioPCMBuffer alloc] initWithPCMFormat:
audioFile.processingFormat frameCapacity: kBufferFrameCapacity];
```

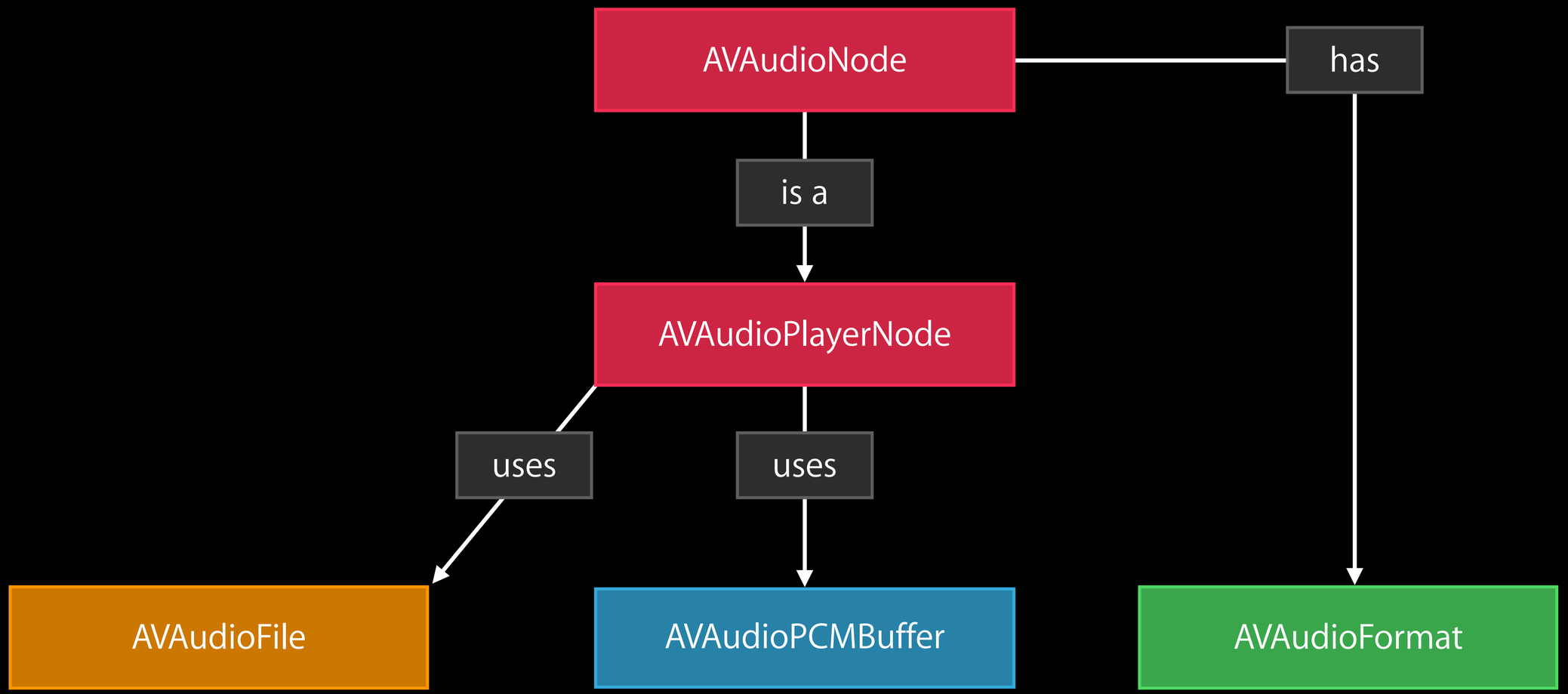
Read the file a buffer at a time to find the loudest sample

```
float loudestSample = 0.0f;
AVAudioFramePosition loudestSamplePosition = 0;



while (audioFile.framePosition < fileLength) {
AVAudioFramePosition readPosition = audioFile.framePosition;
if (![audioFile readIntoBuffer: readBuffer error: &error]) {
  NSLog(@"failed to read audio file: %@", error);
  return NO;
}
if (readBuffer.frameLength == 0)
  break; // finished
```

For each channel, check each audio sample

```
for (AVAudioChannelCount channelIndex = 0;
    channelIndex < readBuffer.format.channelCount; ++channelIndex)
{
  float *channelData = readBuffer.floatChannelData[channelIndex];
  for (AVAudioFrameCount frameIndex = 0;
       frameIndex < readBuffer.frameLength; ++frameIndex)
  {
    float sampleAbsLevel = fabs(channelData[frameIndex]);
    if (sampleAbsLevel > loudestSample)
    {
      loudestSample = sampleAbsLevel;
      loudestSamplePosition = readPosition + frameIndex;
    }
```

With AVAudioEngine Classes



AV Foundation Audio—Summary

Classes

- AVAudioFormat
- AVAudioChannelLayout
- AVAudioPCMBuffer
- AVAudioFile

Use with Core Audio, Audio Toolbox and Audio Unit C APIs

- Via accessors

AVAudioEngine

Summary

MIDI over Bluetooth

Inter-App Audio UI Views

Enhanced AV Foundation audio

- Audio Unit Manager
- AVAudioSession
- Utility classes
- AVAudioEngine

More Information

Filip Iliescu
Graphics and Game Technologies Evangelist
filiescu@apple.com

Apple Developer Forums
http://devforums.apple.com

Related Sessions

• AVAudioEngine in Practice	Marina	Tuesday 10:15AM

Labs

• Audio Lab	Media Lab A	Tuesday 11:30AM
• Audio Lab	Media Lab B	Wednesday 12:45PM

WWDC14