Introducing AirPlay 2
Unlocking multi-room audio
Session 509

David Saracino, AirPlay Engineer
AirPlay Overview

Screen

Audio

Video
AirPlay Overview

Screen

Audio

Video
AirPlay Overview

AirPlay 2

Screen

Audio

Video
AirPlay 2
Audio

Wireless audio
AirPlay 2
Audio

Wireless audio

Multi-room playback
AirPlay 2
Audio

Wireless audio
Multi-room playback
Enhanced buffering
AirPlay 2
Audio

Wireless audio
Multi-room playback
Enhanced buffering
AirPlay 2
Audio

Wireless audio
Multi-room playback
Enhanced buffering
Multi-device control
AirPlay 2
Supported speakers

HomePod
Apple TV*
3rd Party

* Requires Apple TV (4th generation)
AirPlay 2 adoption
Advanced playback scenarios
Availability
AirPlay 2 Adoption
AirPlay 2 Adoption
AirPlay 2 Adoption

Identify as long-form audio
AirPlay 2 Adoption

Identify as long-form audio

Add an AirPlay picker
AirPlay 2 Adoption

Identify as long-form audio

Add an AirPlay picker

Integrate with MediaPlayer framework
AirPlay 2 Adoption

Identify as long-form audio

Add an AirPlay picker

Integrate with MediaPlayer framework

Adopt an AirPlay 2 playback API
Identify as Long-Form Audio

NEW
Identify as Long-Form Audio

Music, podcasts, or audiobooks
Identify as Long-Form Audio

Music, podcasts, or audiobooks

Set AVAudioSession route sharing policy
Identify as Long-Form Audio

Music, podcasts, or audiobooks

Set AVAudioSession route sharing policy

```swift
let audioSession = AVAudioSession.sharedInstance()
try audioSession.setCategory(AVAudioSessionCategoryPlayback,
    mode: AVAudioSessionModeDefault,
    routeSharingPolicy: .longForm)
```
Identify as Long-Form Audio

Music, podcasts, or audiobooks

Set AVAudioSession route sharing policy

```swift
let audioSession = AVAudioSession.sharedInstance()
try audioSession.setCategory(AVAudioSessionCategoryPlayback,
    mode: AVAudioSessionModeDefault,
    routeSharingPolicy: .longForm)
```
Add an AirPlay Picker
Add an AirPlay Picker

Adopt
Add an AirPlay Picker

Adopt

- AVRoutePickerView
Add an AirPlay Picker

Adopt

• AVRoutePickerView
• AVRouteDetector
Integrate with Media Player Framework
Integrate with Media Player Framework

Handle remote media commands

- MPRemoteCommandCenter
Integrate with Media Player Framework

Handle remote media commands
- **MPRemoteCommandCenter**

Display current track info
- **MPNowPlayingInfoCenter**
AirPlay 2 Playback APIs
Audio Buffering
Existing AirPlay
Audio Buffering
Existing AirPlay

Real-time audio stream
Audio Buffering
Existing AirPlay

Real-time audio stream
Speaker adds a small buffer before output
Audio Buffering
Existing AirPlay

Real-time audio stream
Speaker adds a small buffer before output
Works fine for streaming to single speaker
Enhanced Audio Buffering
AirPlay 2
NEW
Enhanced Audio Buffering

AirPlay 2

Large audio buffering capacity on speakers
Enhanced Audio Buffering
AirPlay 2

Large audio buffering capacity on speakers
Faster-than-real-time streaming to speakers
Enhanced Audio Buffering
AirPlay 2

Large audio buffering capacity on speakers

Faster-than-real-time streaming to speakers

Benefits
Enhanced Audio Buffering
AirPlay 2

Large audio buffering capacity on speakers
Faster-than-real-time streaming to speakers

Benefits
• Adds robustness
Enhanced Audio Buffering

AirPlay 2

Large audio buffering capacity on speakers

Faster-than-real-time streaming to speakers

Benefits

• Adds robustness

• More responsive playback
Enhanced Audio Buffering

Adoption

Supported with specific playback APIs
Enhanced Audio Buffering

Adoption

Supported with specific playback APIs

- AVPlayer / AVQueuePlayer
Enhanced Audio Buffering

Adoption

Supported with specific playback APIs

- AVPlayer / AVQueuePlayer
- AVSampleBufferAudioRenderer
  - AVSampleBufferRendererSynchronizer
Enhanced Audio Buffering
AVPlayer / AVQueuePlayer
Enhanced Audio Buffering
AVPlayer / AVQueuePlayer

Client App
Enhanced Audio Buffering
AVPlayer / AVQueuePlayer
Enhanced Audio Buffering
AVPlayer / AVQueuePlayer
Enhanced Audio Buffering

AVPlayer / AVQueuePlayer
Enhanced Audio Buffering
AVPlayer / AVQueuePlayer
Enhanced Audio Buffering
AVPlayer / AVQueuePlayer
Enhanced Audio Buffering

AVPlayer / AVQueuePlayer

Client App

AVQueuePlayer

AVPlayerItem

AVAsset

URL
Enhanced Audio Buffering
AVPlayer / AVQueuePlayer
Enhanced Audio Buffering
AVPlayer / AVQueuePlayer

Client App

SetRate 1

AVQueuePlayer

AVPlayerItem

AVAsset

URL
Enhanced Audio Buffering
AVPlayer / AVQueuePlayer

Audio Data
Client App
AVQueuePlayer
AVPlayerItem
AVAsset
URL
Enhanced Audio Buffering
AVPlayer / AVQueuePlayer

Client App

Audio Data
AVQueuePlayer
AVPlayerItem
AVAsset
URL
Enhanced Audio Buffering
AVPlayer / AVQueuePlayer

Audio Data → Client App

Audio Data

AVQueuePlayer

AVPlayerItem
AVAsset
URL
Enhanced Audio Buffering
AVPlayer / AVQueuePlayer
Enhanced Audio Buffering
AVPlayer / AVQueuePlayer
Enhanced Audio Buffering
AVPlayer / AVQueuePlayer

Client App

AVQueuePlayer

Audio Data

AVPlayerItem

AVAsset

URL
Enhanced Audio Buffering
AVSampleBufferAudioRenderer / AVSampleBufferRenderSynchronizer

NEW
Enhanced Audio Buffering
AVSampleBufferAudioRenderer / AVSampleBufferRendererSynchronizer

Your app has additional responsibilities
Enhanced Audio Buffering
AVSampleBufferAudioRenderer / AVSampleBufferRenderSynchronizer

Your app has additional responsibilities
• Sourcing and parsing the content
Enhanced Audio Buffering
AVSampleBufferAudioRenderer / AVSampleBufferRenderSynchronizer

Your app has additional responsibilities

• Sourcing and parsing the content
• Providing raw audio buffers for rendering
Enhanced Audio Buffering
AVSampleBufferAudioRenderer / AVSampleBufferRenderSynchronizer
Enhanced Audio Buffering
AVSampleBufferAudioRenderer / AVSampleBufferRendererSynchronizer

Client App
Enhanced Audio Buffering
AVSampleBufferAudioRenderer / AVSampleBufferRenderSynchronizer
Enhanced Audio Buffering
AVSampleBufferAudioRenderer / AVSampleBufferRenderSynchronizer

Client App

AudioRenderer

Synchronizer
Enhanced Audio Buffering
AVSampleBufferAudioRenderer / AVSampleBufferRendererSynchronizer
Enhanced Audio Buffering
AVSampleBufferAudioRenderer / AVSampleBufferRenderSynchronizer

Client App

Wants more data

AudioRenderer

Synchronizer
Enhanced Audio Buffering
AVSampleBufferAudioRenderer / AVSampleBufferRendererSynchronizer

Client App

Wants more data

AudioRenderer

Synchronizer
Enhanced Audio Buffering
AVSampleBufferAudioRenderer / AVSampleBufferRenderSynchronizer

Audio Data

Client App

AudioRenderer

Synchronizer
Enhanced Audio Buffering
AVSampleBufferAudioRenderer / AVSampleBufferRendererSynchronizer

Audio Data

Client App

Audio Data

AudioRenderer

Synchronizer
Enhanced Audio Buffering
AVSampleBufferAudioRenderer / AVSampleBufferRenderSynchronizer

Client App

Audio Data

AudioRenderer

Synchronizer
Enhanced Audio Buffering

AVSampleBufferAudioRenderer / AVSampleBufferRenderSynchronizer

Client App

AudioData

AudioRenderer

Synchronizer

SetRate 1
Enhanced Audio Buffering
AVSampleBufferAudioRenderer / AVSampleBufferRenderSynchronizer

Client App

SetRate 1

Audio Data

AudioRenderer

Synchronizer
Enhanced Audio Buffering

AVSampleBufferAudioRenderer / AVSampleBufferRenderSynchronizer

Client App

AudioRenderer

Synchronizer

Audio Data

Audio Data

Audio Data
Enhanced Audio Buffering
AVSampleBufferAudioRenderer / AVSampleBufferRenderSynchronizer

Client App

AudioRenderer

Synchronizer

Audio Data
Demo

AirPlay 2 with enhanced buffering
Advanced Playback Scenarios
AVSampleBufferAudioRenderer

David Saracino, AirPlay Engineer
AVSampleBufferAudioRenderer

Audio buffer levels

Seek

Play queues

Supported audio formats

Video synchronization
Audio Buffer Levels

AVSampleBufferAudioRenderer
Audio Buffer Levels
AVSampleBufferAudioRenderer

The amount of data requested will vary depending on the current route
The amount of data requested will vary depending on the current route.
Audio Buffer Levels
AVSampleBufferAudioRenderer

The amount of data requested will vary depending on the current route
Audio Buffer Levels
AVSampleBufferAudioRenderer

Audio Data to Play

Source

Renderer

Playhead
Audio Buffer Levels
AVSampleBufferAudioRenderer

Local Playback: enqueue just a few seconds ahead of the Playhead
Audio Buffer Levels
AVSampleBufferAudioRenderer

Local Playback: enqueue just a few seconds ahead of the Playhead
Audio Buffer Levels

AVSampleBufferAudioRenderer

Local Playback: enqueue just a few seconds ahead of the Playhead

Source

Audio Data to Play

Renderer
Audio Buffer Levels
AVSampleBufferAudioRenderer

Source

Audio Data to Play

Renderer

Last Enqueued Sample
Audio Buffer Levels
AVSampleBufferAudioRenderer

AirPlay 2 Speaker: enqueue up to multiple minutes of the Playhead
Audio Buffer Levels
AVSampleBufferAudioRenderer

AirPlay 2 Speaker: enqueue up to multiple minutes of the Playhead
Audio Buffer Levels
AVSampleBufferAudioRenderer

AirPlay 2 Speaker: enqueue up to multiple minutes of the Playhead
Audio Buffer Levels
AVSampleBufferAudioRenderer

Requested data amount varies by audio route
Audio Buffer Levels
AVSampleBufferAudioRenderer

Requested data amount varies by audio route
<table>
<thead>
<tr>
<th>Local</th>
<th>Seconds</th>
</tr>
</thead>
</table>

Requested data amount varies by audio route
## Audio Buffer Levels

**AVSampleBufferAudioRenderer**

Requested data amount varies by audio route

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Local</strong></td>
<td><strong>Seconds</strong></td>
</tr>
<tr>
<td><strong>Bluetooth</strong></td>
<td><strong>Seconds</strong></td>
</tr>
</tbody>
</table>
Audio Buffer Levels
AVSampleBufferAudioRenderer

Requested data amount varies by audio route

<table>
<thead>
<tr>
<th>Route</th>
<th>Seconds</th>
</tr>
</thead>
<tbody>
<tr>
<td>Local</td>
<td></td>
</tr>
<tr>
<td>Bluetooth</td>
<td></td>
</tr>
<tr>
<td>AirPlay</td>
<td></td>
</tr>
</tbody>
</table>
Audio Buffer Levels
AVSampleBufferAudioRenderer

Requested data amount varies by audio route

<table>
<thead>
<tr>
<th>Audio Route</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Local</td>
<td>Seconds</td>
</tr>
<tr>
<td>Bluetooth</td>
<td>Seconds</td>
</tr>
<tr>
<td>AirPlay</td>
<td>Seconds</td>
</tr>
<tr>
<td>AirPlay 2</td>
<td>Minutes</td>
</tr>
</tbody>
</table>
Audio Buffer Levels
AVSampleBufferAudioRenderer

Requested data amount varies by audio route

<table>
<thead>
<tr>
<th>Route</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Local</td>
<td>Seconds</td>
</tr>
<tr>
<td>Bluetooth</td>
<td>Seconds</td>
</tr>
<tr>
<td>AirPlay</td>
<td>Seconds</td>
</tr>
<tr>
<td>AirPlay 2</td>
<td>Minutes</td>
</tr>
</tbody>
</table>
Seek
AVSampleBufferAudioRenderer
Seek
AVSampleBufferAudioRenderer
Seek
AVSampleBufferAudioRenderer

Manually changing Playhead location
Seek
AVSampleBufferAudioRenderer

Manually changing Playhead location
Seek
AVSampleBufferAudioRenderer

Manually changing Playhead location

Source
Audio Data to Play

Renderer

Playhead
Seek
AVSampleBufferAudioRenderer

Source

Audio Data to Play

Renderer

Playhead
Seek

AVSampleBufferAudioRenderer

Source

Audio Data to Play

Renderer

Playhead
Seek
AVSampleBufferAudioRenderer

Source
Audio Data to Play

Renderer
Playhead
Seek
AVSampleBufferAudioRenderer

Source

Audio Data to Play

Renderer

Playhead

Playhead
Seek
AVSampleBufferAudioRenderer
Seek

AVSampleBufferAudioRenderer

Source

Audio Data to Play

Renderer

Playhead
Seek
AVSampleBufferAudioRenderer

Source
Audio Data to Play

Flush
Playhead

Renderer
Seek

AVSampleBufferAudioRenderer

Source: Audio Data to Play

Renderer

Playhead
Seek
AVSampleBufferAudioRenderer

Source
Audio Data to Play

Renderer

Playhead
Seek
AVSampleBufferAudioRenderer

Source
Audio Data to Play

Renderer
// Seek - AVSampleBufferAudioRenderer

func seek(toMediaTime mediaTime: CMTime) {

    renderSynchronizer.setRate(0.0, time: kCMTimeZero)
    audioRenderer.stopRequestingMediaData()

    audioRenderer.flush()

    myPrepareSampleGenerationForMediaTime(mediaTime)

    audioRenderer.requestMediaDataWhenReady(on: mySerializationQueue) {
        // ...
    }

    renderSynchronizer.setRate(1.0, time: mediaTime)
}
func seek(toMediaTime mediaTime: CMTime) {
    renderSynchronizer.setRate(0.0, time: kCMTimeZero)
    audioRenderer.stopRequestingMediaData()

    audioRenderer.flush()

    myPrepareSampleGenerationForMediaTime(mediaTime)

    audioRenderer.requestMediaDataWhenReady(on: mySerializationQueue) {
        // ...
    }

    renderSynchronizer.setRate(1.0, time: mediaTime)
}
/* Seek - AVSampleBufferAudioRenderer */

```swift
func seek(toMediaTime mediaTime: CMTTime) {
    renderSynchronizer.setRate(0.0, time: kCMTimeZero)
    audioRenderer.stopRequestingMediaData()

    audioRenderer.flush()

    myPrepareSampleGenerationForMediaTime(mediaTime)

    audioRenderer.requestMediaDataWhenReady(on: mySerializationQueue) {
        // ...
    }

    renderSynchronizer.setRate(1.0, time: mediaTime)
}
```
// Seek - AVSampleBufferAudioRenderer

func seek(toMediaTime mediaTime: CMTime) {

    renderSynchronizer.setRate(0.0, time: kCMTimeZero)
    audioRenderer.stopRequestingMediaData()

    audioRenderer.flush()

    myPrepareSampleGenerationForMediaTime(mediaTime)

    audioRenderer.requestMediaDataWhenReady(on: mySerializationQueue) {
        // ...
    }

    renderSynchronizer.setRate(1.0, time: mediaTime)
}
// Seek - AVSampleBufferAudioRenderer

func seek(toMediaTime mediaTime: CMTime) {

    renderSynchronizer.setRate(0.0, time: kCMTimeZero)
    audioRenderer.stopRequestingMediaData()

    audioRenderer.flush()

    myPrepareSampleGenerationForMediaTime(mediaTime)

    audioRenderer.requestMediaDataWhenReady(on: mySerializationQueue) {
        // ...
    }

    renderSynchronizer.setRate(1.0, time: mediaTime)
}

// Seek - AVSampleBufferAudioRenderer

func seek(toMediaTime mediaTime: CMTime) {

    renderSynchronizer.setRate(0.0, time: kCMTimeZero)
    audioRenderer.stopRequestingMediaData()

    audioRenderer.flush()

    myPrepareSampleGenerationForMediaTime(mediaTime)

    audioRenderer.requestMediaDataWhenReady(on: mySerializationQueue) {
        // ...
    }

    renderSynchronizer.setRate(1.0, time: mediaTime)
}
Play Queues
AVSampleBufferAudioRenderer
Play Queues

Item 1
AirPlay Too

Item 2
AirPlay Too

Item 3
AirPlay Too

Item 4
AirPlay Too
Play Queues

Queue

Item 1
AirPlay Too

Item 2
AirPlay Too

Item 3
AirPlay Too

Item 4
AirPlay Too
Play Queues

Queue

- Item 1
- Item 2
- Item 3
Play Queues
Timelines

Queue

Item 1

Item 2

Item 3
Play Queues

Timelines

Queue

Item 1

Item 2

Item 3
Play Queues

Timelines

Queue

Renderer

0

100

0

100

0

Item 1

Item 2

Item 3
Play Queues

Timelines

Queue

Renderer

Item 1

Item 2

Item 3

Continuous Timeline
Play Queues
Enqueuing Audio Renderer
Play Queues
Enqueuing Audio Renderer

Queue
- Item 1
- Item 2
- Item 3

Renderer

Last Enqueued Sample
Play Queues
Enqueuing Audio Renderer

Queue
Item 1
Item 2
Item 3

Renderer
Play Queues
Deep Audio Buffer Levels

Queue
- Item 1
- Item 2
- Item 3

Renderer

Playhead

Last Enqueued Sample
Play Queues
Deep Audio Buffer Levels

Queue
- Item 1
- Item 2
- Item 3

Renderer
Play Queues

Editing

Queue
- Item 1

Renderer

Playhead

Last Enqueued Sample

Item 3
Play Queues

Editing

Queue

Item 1

Item 3

Item 4

Renderer

Playhead

Last Enqueued Sample
Play Queues

Editing during playback

Queue

Item 1

Item 2

Item 3

Renderer

Playhead

Last Enqueued Sample
Play Queues

Editing during playback

Queue

| Item 1 | Item 2 | Item 3 |

Renderer

Playhead

Last Enqueued Sample
Play Queues

Editing during playback

Queue

Item 1

Item 2

Item 3

Renderer

Playhead

Last Enqueued Sample
Play Queues
Editing during playback

Queue:
- Item 1

Renderer:
- Playhead
- Last Enqueued Sample

Item 3
Play Queues

Editing during playback

Queue

Item 1

Item 3

Item 4

Renderer

Playhead

Last Enqueued Sample
Play Queues
Editing during playback

Queue
- Item 1
- Item 3
- Item 4

Renderer
- Item 2’s Data!

Playhead
Last Enqueued Sample
Play Queues
Replacing incorrect renderer data
Play Queues
Replacing incorrect renderer data

Queue
Item 1
Item 3
Item 4

Renderer

Flush from Source Time
Play Queues
Replacing incorrect renderer data
Play Queues
Replacing incorrect renderer data
Play Queues
Replacing incorrect renderer data
Play Queues
Replacing incorrect renderer data

Queue
- Item 1
- Item 3
- Item 4

Renderer
Flushing from a Source Time
1. Stop enqueueing audio data
Flushing from a Source Time

1. Stop enqueueing audio data

2. `flush(fromSourceTime:)`
Flush from a Source Time

1. Stop enqueueing audio data

2. `flush(fromSourceTime:)`

3. Wait for the callback
Flushing from a Source Time
Gotchas
Flush may fail!
Flush may fail!

• Source time is too close to playhead
Flush may fail!

- Source time is too close to playhead

Wait for the callback!
func performFlush(fromSourceTime sourceTime: CMTime) {

    audioRenderer.stopRequestingMediaData()

    // App-specific logic to ensure no more media data is enqueued

    audioRenderer.flush(fromSourceTime: sourceTime) { (flushSucceeded) in
        if flushSucceeded {
            self.myPrepareSampleGenerationForMediaTime(sourceTime)
            audioRenderer.requestMediaDataWhenReady(on: mySerializationQueue) { /*...*/ }
        }
        else {
            // Flush and interrupt playback
        }
    }
}
func performFlush(fromSourceTime sourceTime: CMTime) {

    audioRenderer.stopRequestingMediaData()

    // App-specific logic to ensure no more media data is enqueued

    audioRenderer.flush(fromSourceTime: sourceTime) { (flushSucceeded) in
        if flushSucceeded {
            self.myPrepareSampleGenerationForMediaTime(sourceTime)
            audioRenderer.requestMediaDataWhenReady(on: mySerializationQueue) { /*...*/ }
        }
        else {
            // Flush and interrupt playback
        }
    }
}
// FlushFromSourceTime - AVSampleBufferAudioRenderer

func performFlush(fromSourceTime sourceTime: CMTime) {

    audioRenderer.stopRequestingMediaData()

    // App-specific logic to ensure no more media data is enqueued

    audioRenderer.flush(fromSourceTime: sourceTime) { (flushSucceeded) in
        if flushSucceeded {
            self.myPrepareSampleGenerationForMediaTime(sourceTime)
            audioRenderer.requestMediaDataWhenReady(on: mySerializationQueue) { /*...*/ }
        }
        else {
            // Flush and interrupt playback
        }
    }
}
// FlushFromSourceTime - AVSampleBufferAudioRenderer

func performFlush(fromSourceTime sourceTime: CMTime) {

    audioRenderer.stopRequestingMediaData()

    // App-specific logic to ensure no more media data is enqueued

    audioRenderer.flush(fromSourceTime: sourceTime) { (flushSucceeded) in
        if flushSucceeded {
            self.myPrepareSampleGenerationForMediaTime(sourceTime)
            audioRenderer.requestMediaDataWhenReady(on: mySerializationQueue) {
                /*...*/
            }
        } else {
            // Flush and interrupt playback
        }
    }
}
func performFlush(fromSourceTime sourceTime: CMTime) {
    audioRenderer.stopRequestingMediaData()

    // App-specific logic to ensure no more media data is enqueued
    audioRenderer.flush(fromSourceTime: sourceTime) { (flushSucceeded) in
        if flushSucceeded {
            self.myPrepareSampleGenerationForMediaTime(sourceTime)
            audioRenderer.requestMediaDataWhenReady(on: mySerializationQueue) { /*...*/ }
        }
        else {
            // Flush and interrupt playback
        }
    }
}
func performFlush(fromSourceTime sourceTime: CMTime) {

    audioRenderer.stopRequestingMediaData()

    // App-specific logic to ensure no more media data is enqueued

    audioRenderer.flush(fromSourceTime: sourceTime) { (flushSucceeded) in
        if flushSucceeded {
            self.myPrepareSampleGenerationForMediaTime(sourceTime)
            audioRenderer.requestMediaDataWhenReady(on: mySerializationQueue) { /*...*/ }
        }
    }
    else {
        // Flush and interrupt playback
    }
}
Audio Format Support
AVSampleBufferAudioRenderer
Supported Audio Formats
AVSampleBufferAudioRenderer
Supported Audio Formats
AVSampleBufferAudioRenderer

All platform-supported audio formats
Supported Audio Formats
AVSampleBufferAudioRenderer

All platform-supported audio formats
• e.g. LPCM, AAC, mp3, or ALAC
Supported Audio Formats
AVSampleBufferAudioRenderer

All platform-supported audio formats
• e.g. LPCM, AAC, mp3, or ALAC
• e.g. 44.1 kHZ or 48 kHZ
Supported Audio Formats
AVSampleBufferAudioRenderer

All platform-supported audio formats
• e.g. LPCM, AAC, mp3, or ALAC
• e.g. 44.1 kHZ or 48 kHZ
• various bit depths
Supported Audio Formats
AVSampleBufferAudioRenderer

All platform-supported audio formats
• e.g. LPCM, AAC, mp3, or ALAC
• e.g. 44.1 kHZ or 48 kHZ
• various bit depths

Mixed formats may be enqueued
Supported Audio Formats
AVSampleBufferAudioRenderer

All platform-supported audio formats
• e.g. LPCM, AAC, mp3, or ALAC
• e.g. 44.1 kHZ or 48 kHZ
• various bit depths

Mixed formats may be enqueued

Renderer
  AAC @ 44.1kHz  MP3 @ 48kHz  16bit ALAC @ 48kHz
Preferred Audio Formats

AVSampleBufferAudioRenderer
Preferred Audio Formats

AVSampleBufferAudioRenderer

Original audio format (decrypted)
Preferred Audio Formats
AVSampleBufferAudioRenderer

Original audio format (decrypted)

Interleaved channel formats
Preferred Audio Formats

AVSampleBufferAudioRenderer

Original audio format (decrypted)

Interleaved channel formats

1-2 seconds of audio per CMSampleBuffer
Video Synchronization
AVSampleBufferDisplayLayer
Video Synchronization

Client App

AudioRenderer  Synchronizer
Video Synchronization

AVSampleBufferDisplayLayer

Client App

AudioRenderer

Synchronizer
Video Synchronization
AVSampleBufferDisplayLayer

Client App

AudioRenderer

Synchronizer
Video Synchronization
AVSampleBufferDisplayLayer
Video Synchronization

AVSampleBufferDisplayLayer

Client App

AudioRenderer

Synchronizer

DisplayLayer

NEW
Video Synchronization
AVSampleBufferDisplayLayer

Client App

Audio Data

AudioRenderer

Synchronizer

DisplayLayer
Video Synchronization
AVSampleBufferDisplayLayer

Client App

AudioRenderer

Synchronizer

DisplayLayer

Audio Data
Video Synchronization
AVSampleBufferDisplayLayer

Client App

AudioRenderer
Audio Data

Synchronizer

DisplayLayer

Video Data

NEW
Video Synchronization
AVSampleBufferDisplayLayer

Client App

AudioRenderer
Audio Data

Synchronizer

DisplayLayer
Video Data

NEW
Video Synchronization

AVSampleBufferDisplayLayer

Client App

AudioRenderer

Synchronizer

DisplayLayer

Audio Data

Video Data

SetRate 1

NEW
Video Synchronization
AVSampleBufferDisplayLayer

Client App

AudioRenderer

Synchronizer

DisplayLayer

Audio Data

Video Data

NEW
Availability of AirPlay 2
## Availability of AirPlay 2

<table>
<thead>
<tr>
<th>APIs and enhanced buffering</th>
<th>Beta 1</th>
</tr>
</thead>
</table>


Availability of AirPlay 2

APIs and enhanced buffering

Beta 1
## Availability of AirPlay 2

<table>
<thead>
<tr>
<th>Feature</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>APIs and enhanced buffering</td>
<td>Beta 1</td>
</tr>
<tr>
<td>Multi-room audio</td>
<td>Upcoming Beta</td>
</tr>
</tbody>
</table>
## Availability of AirPlay 2

<table>
<thead>
<tr>
<th>Feature</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>APIs and enhanced buffering</td>
<td>Beta 1</td>
</tr>
<tr>
<td>Multi-room audio</td>
<td>Upcoming Beta</td>
</tr>
<tr>
<td>Available to users</td>
<td>Customer Release</td>
</tr>
</tbody>
</table>
Summary
AirPlay 2 introduces many new features for audio
Summary

AirPlay 2 introduces many new features for audio

Long-form audio applications can enable AirPlay 2 with a few steps
Summary

AirPlay 2 introduces many new features for audio

Long-form audio applications can enable AirPlay 2 with a few steps

AirPlay 2 adoption can begin today
More Information

## Related Sessions

<table>
<thead>
<tr>
<th>Session</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>What's New in Audio</td>
<td>WWDC 2017</td>
</tr>
<tr>
<td>Introducing MusicKit</td>
<td>WWDC 2017</td>
</tr>
<tr>
<td>Labs</td>
<td></td>
</tr>
<tr>
<td>--------------</td>
<td></td>
</tr>
<tr>
<td><strong>AirPlay Lab</strong></td>
<td><strong>Technology Lab A</strong></td>
</tr>
</tbody>
</table>