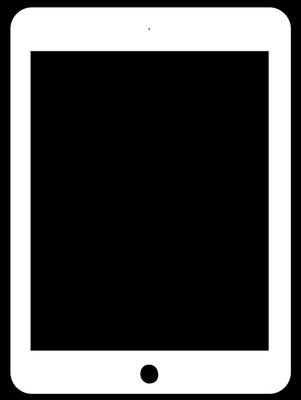
Introducing AirPlay 2

Unlocking multi-room audio

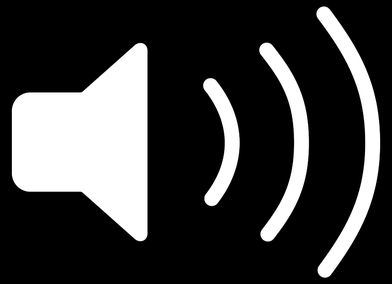
Session 509

David Saracino, AirPlay Engineer

AirPlay Overview



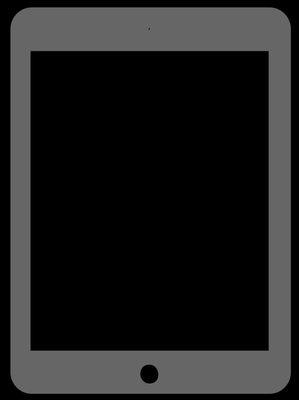
Screen



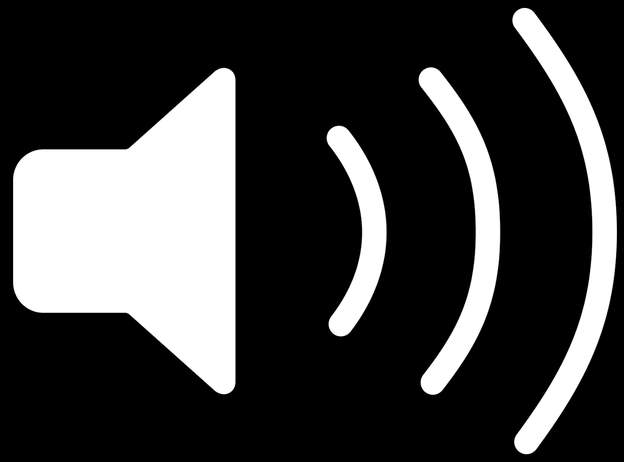
Audio

Video

AirPlay Overview



Screen

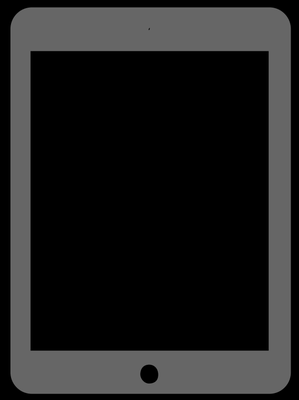


Audio

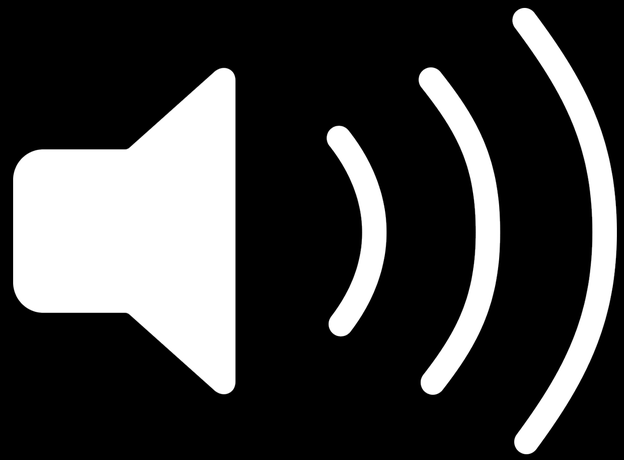
Video

AirPlay Overview

AirPlay 2



Screen



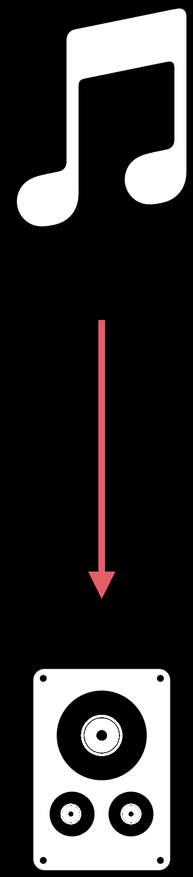
Audio

Video

AirPlay 2

Audio

Wireless audio



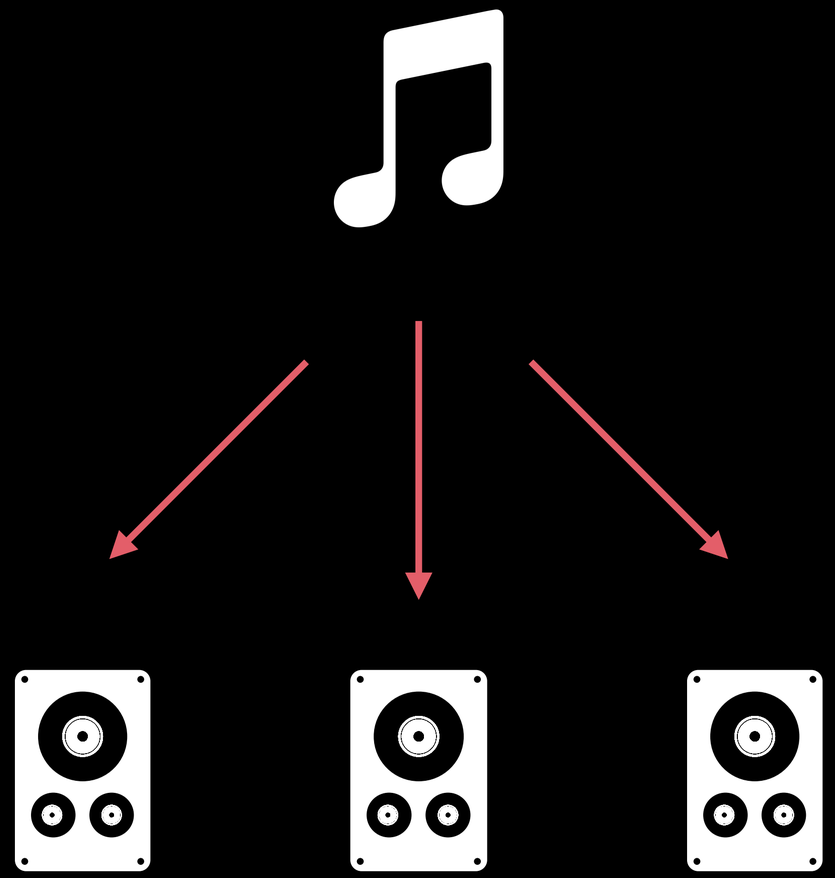
AirPlay 2

Audio

NEW

Wireless audio

Multi-room playback



AirPlay 2

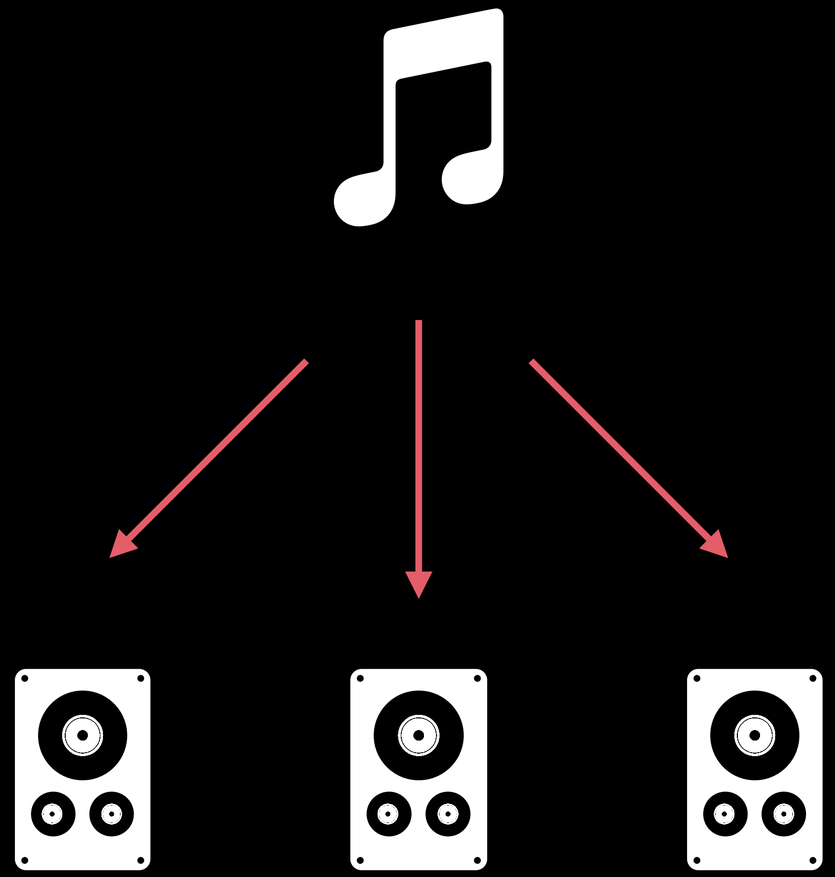
Audio

NEW

Wireless audio

Multi-room playback

Enhanced buffering



AirPlay 2

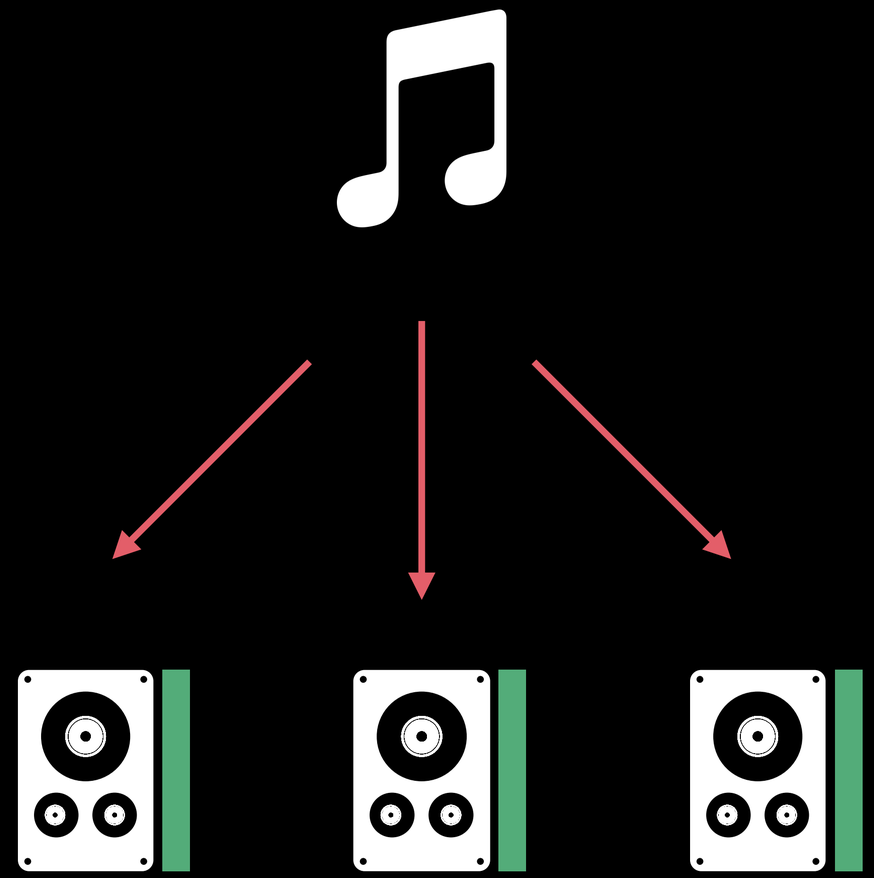
Audio

NEW

Wireless audio

Multi-room playback

Enhanced buffering

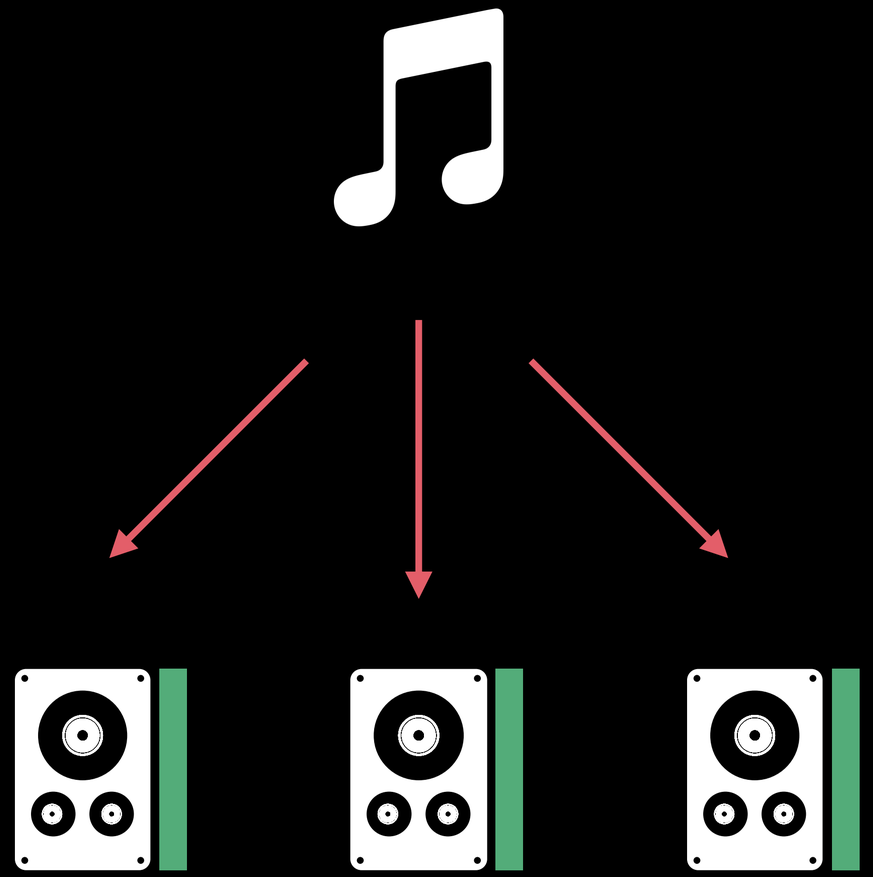


AirPlay 2

Audio

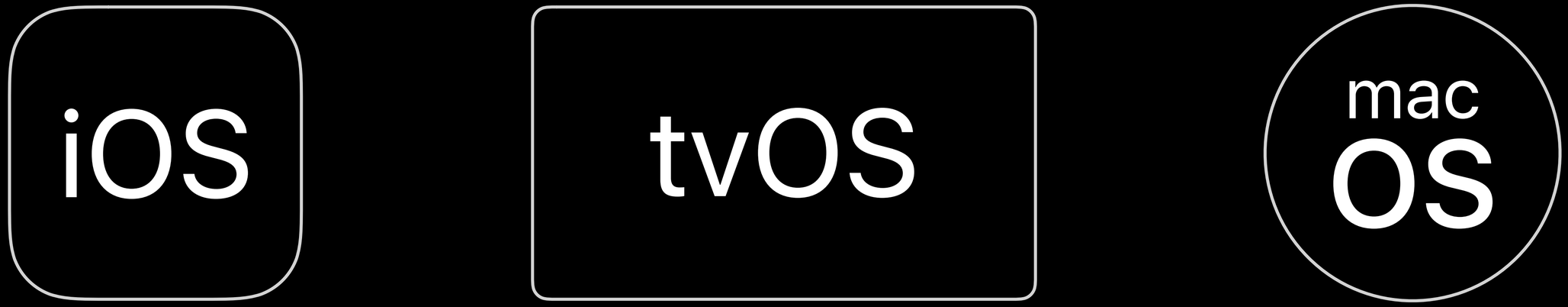
NEW

- Wireless audio
- Multi-room playback
- Enhanced buffering
- Multi-device control



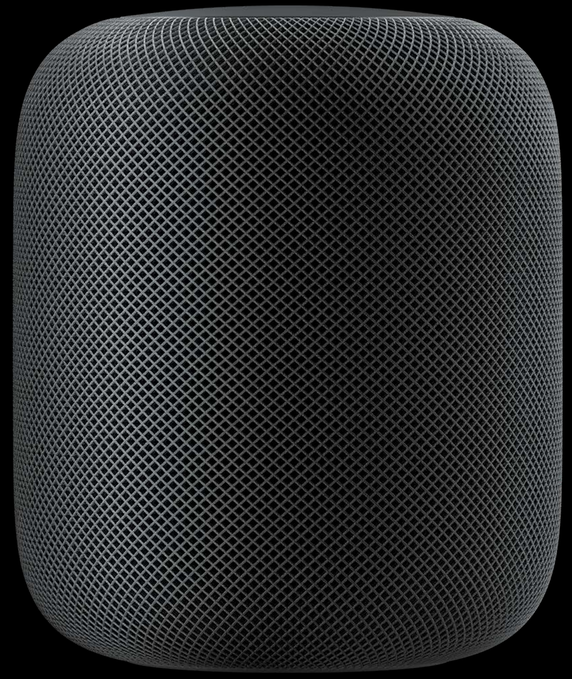
AirPlay 2

Supported platforms



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

NEW

Music, podcasts, or audiobooks

Identify as Long-Form Audio

NEW

Music, podcasts, or audiobooks

Set `AVAudioSession` route sharing policy

Identify as Long-Form Audio

NEW

Music, podcasts, or audiobooks

Set `AVAudioSession` route sharing policy

```
let audioSession = AVAudioSession.sharedInstance()
try audioSession.setCategory(AVAudioSessionCategoryPlayback,
                             mode: AVAudioSessionModeDefault,
                             routeSharingPolicy: .longForm )
```

Identify as Long-Form Audio

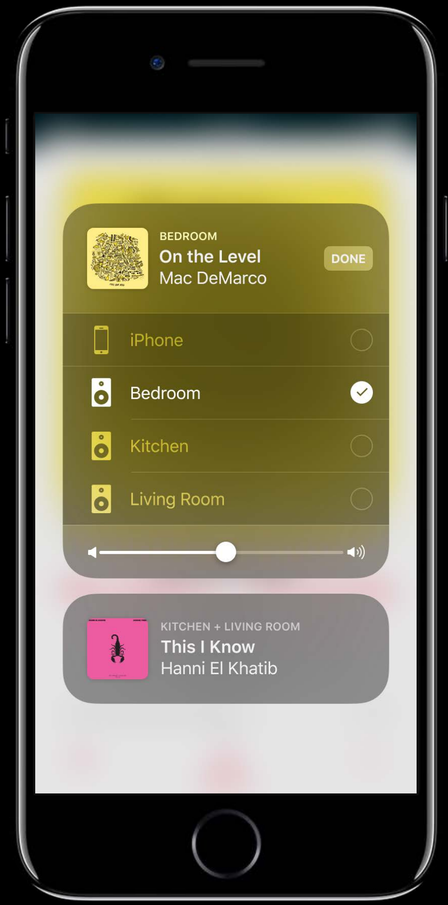
NEW

Music, podcasts, or audiobooks

Set `AVAudioSession` route sharing policy

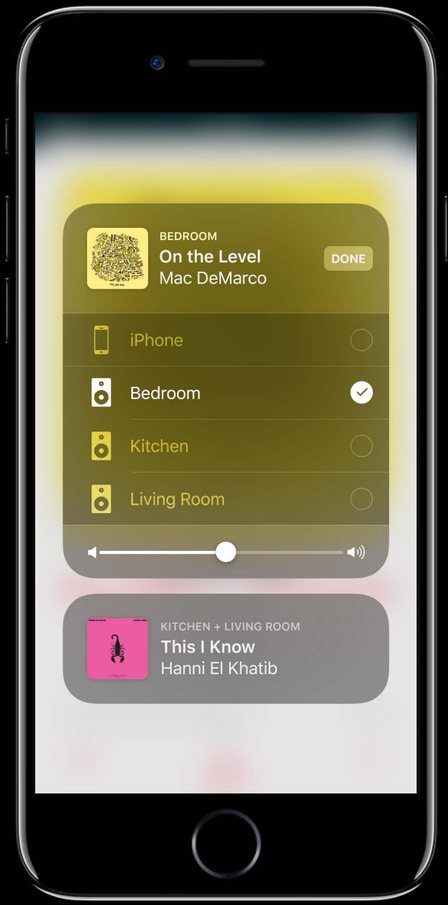
```
let audioSession = AVAudioSession.sharedInstance()
try audioSession.setCategory(AVAudioSessionCategoryPlayback,
                             mode: AVAudioSessionModeDefault,
                             routeSharingPolicy: .longForm )
```

Add an AirPlay Picker



Add an AirPlay Picker

Adopt

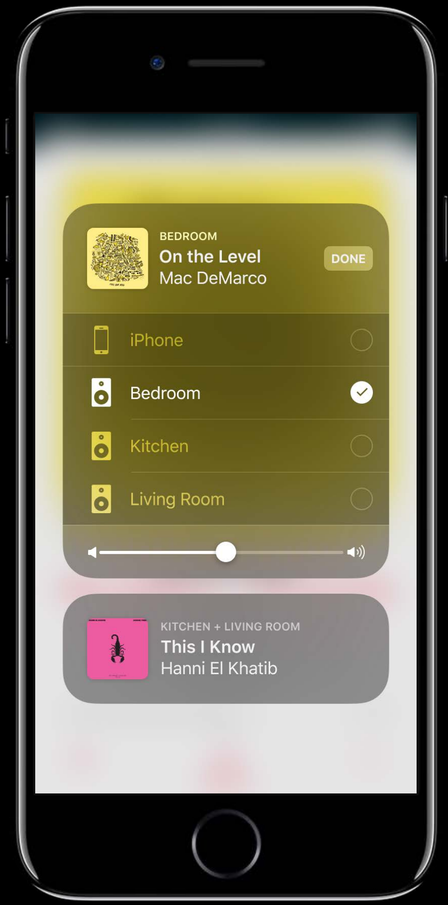


Add an AirPlay Picker

NEW

Adopt

- `AVRoutePickerView`

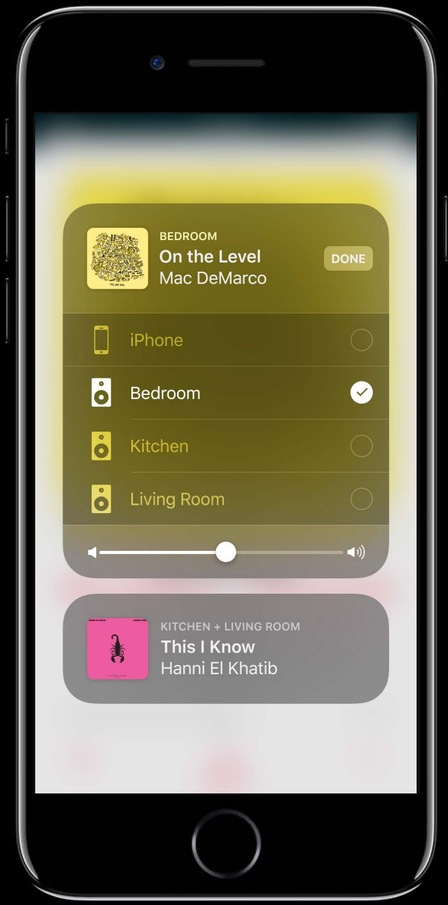


Add an AirPlay Picker

NEW

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

NEW

Large audio buffering capacity on speakers

Enhanced Audio Buffering

AirPlay 2

NEW

Large audio buffering capacity on speakers

Faster-than-real-time streaming to speakers

Enhanced Audio Buffering

AirPlay 2

NEW

Large audio buffering capacity on speakers

Faster-than-real-time streaming to speakers

Benefits

Enhanced Audio Buffering

AirPlay 2

NEW

Large audio buffering capacity on speakers

Faster-than-real-time streaming to speakers

Benefits

- Adds robustness

Enhanced Audio Buffering

AirPlay 2

NEW

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

NEW

Supported with specific playback APIs

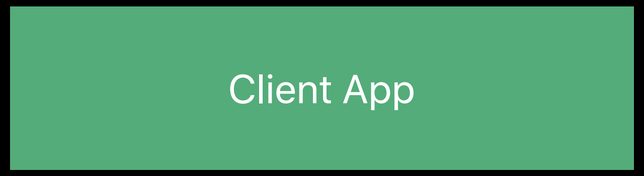
- `AVPlayer / AVQueuePlayer`
- `AVSampleBufferAudioRenderer`
  - `AVSampleBufferRenderSynchronizer`

Enhanced Audio Buffering

AVPlayer / AVQueuePlayer

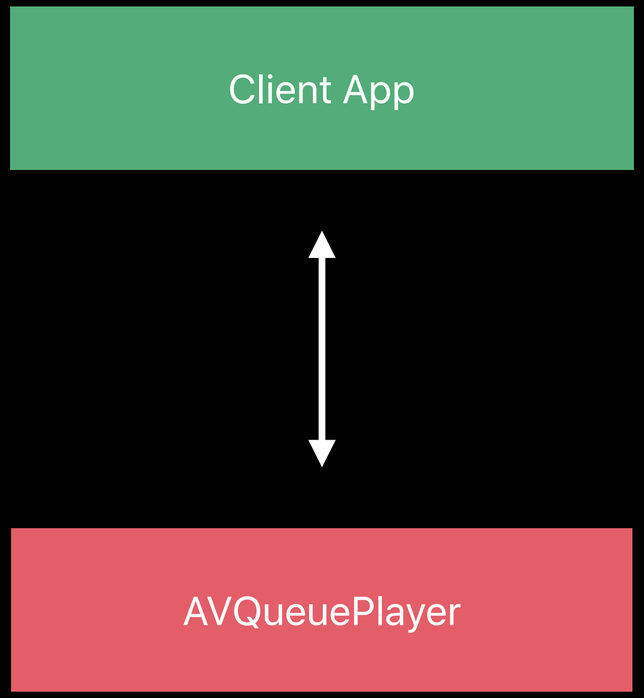
Enhanced Audio Buffering

AVPlayer / AVQueuePlayer



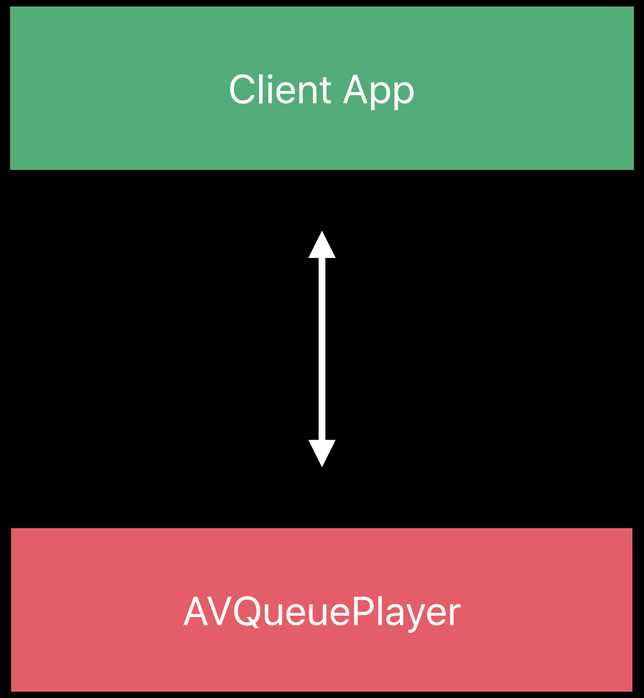
Enhanced Audio Buffering

AVPlayer / AVQueuePlayer



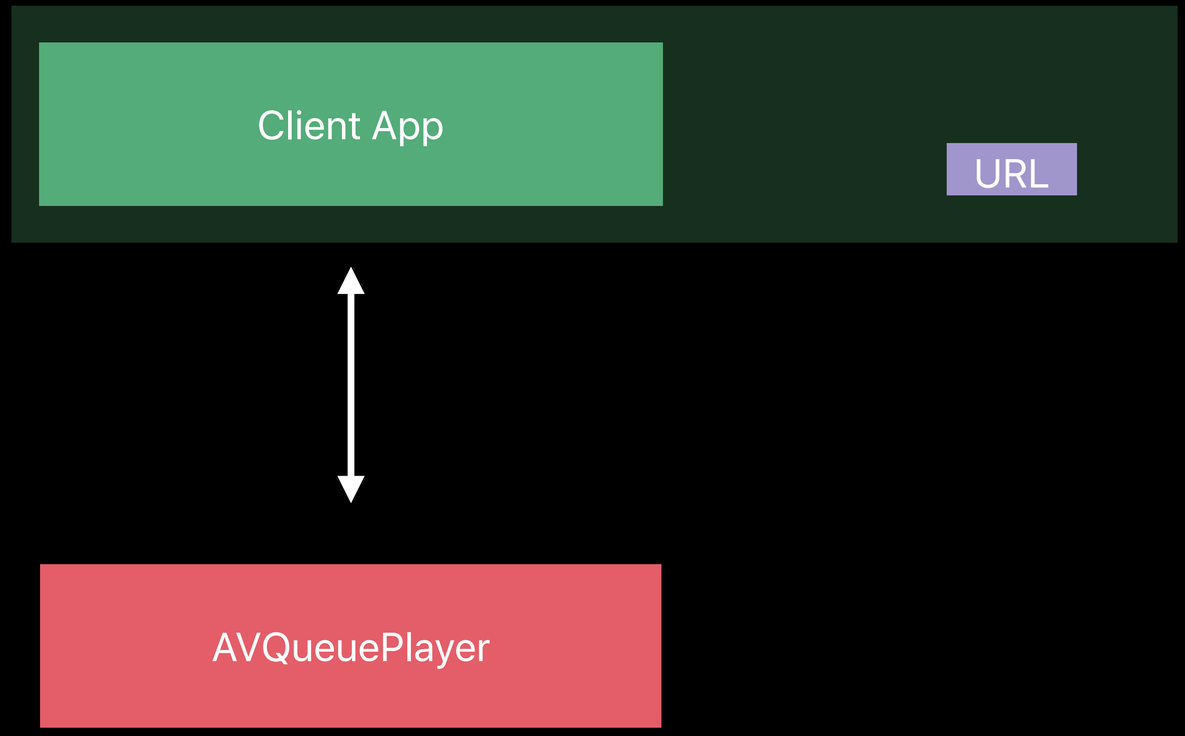
Enhanced Audio Buffering

AVPlayer / AVQueuePlayer



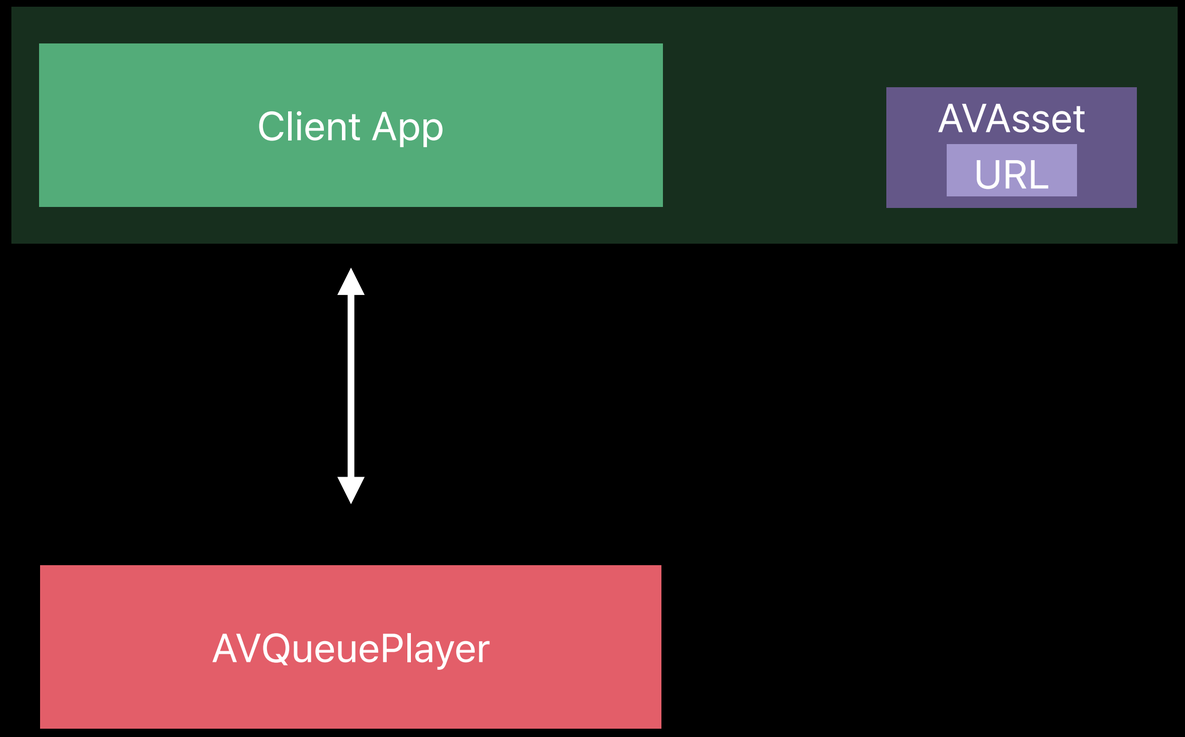
Enhanced Audio Buffering

AVPlayer / AVQueuePlayer



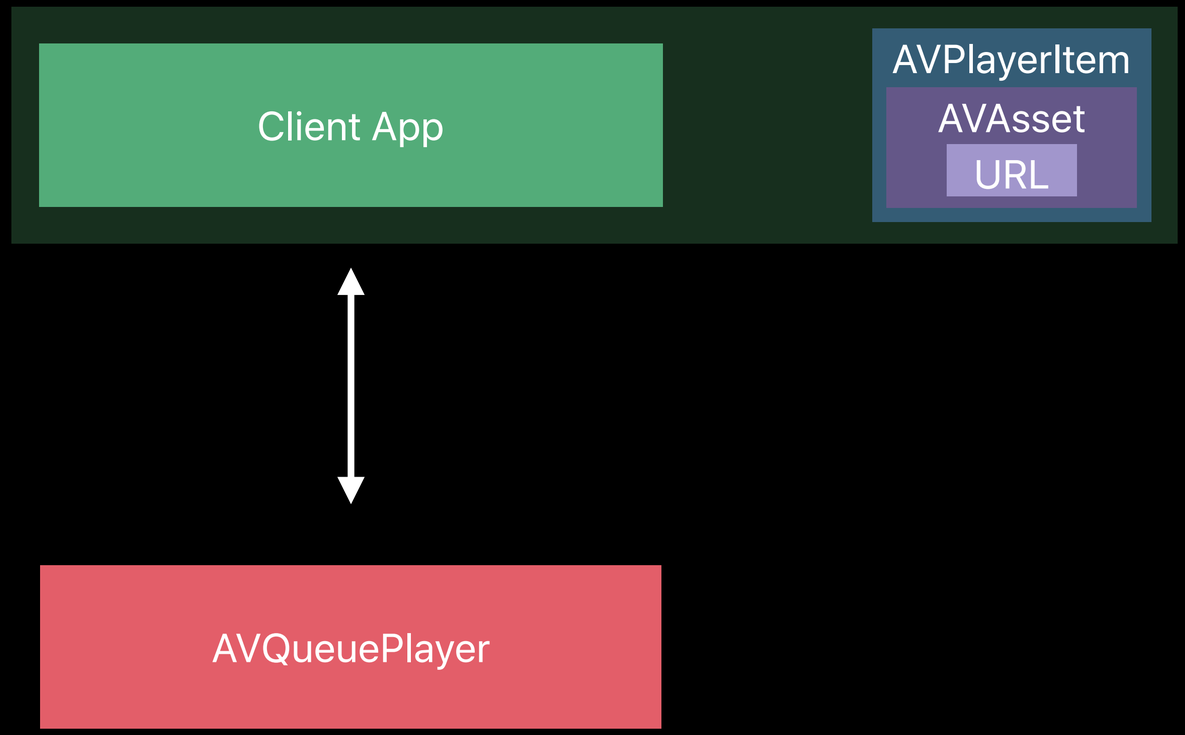
Enhanced Audio Buffering

AVPlayer / AVQueuePlayer



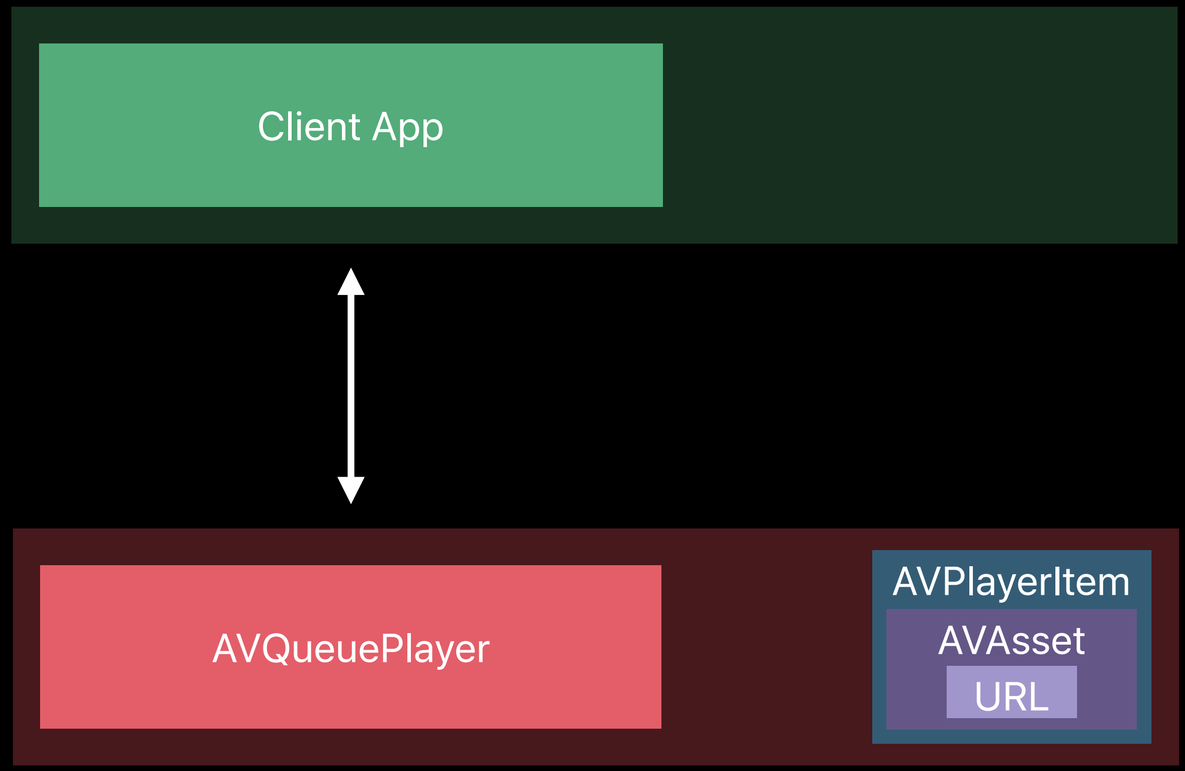
Enhanced Audio Buffering

AVPlayer / AVQueuePlayer



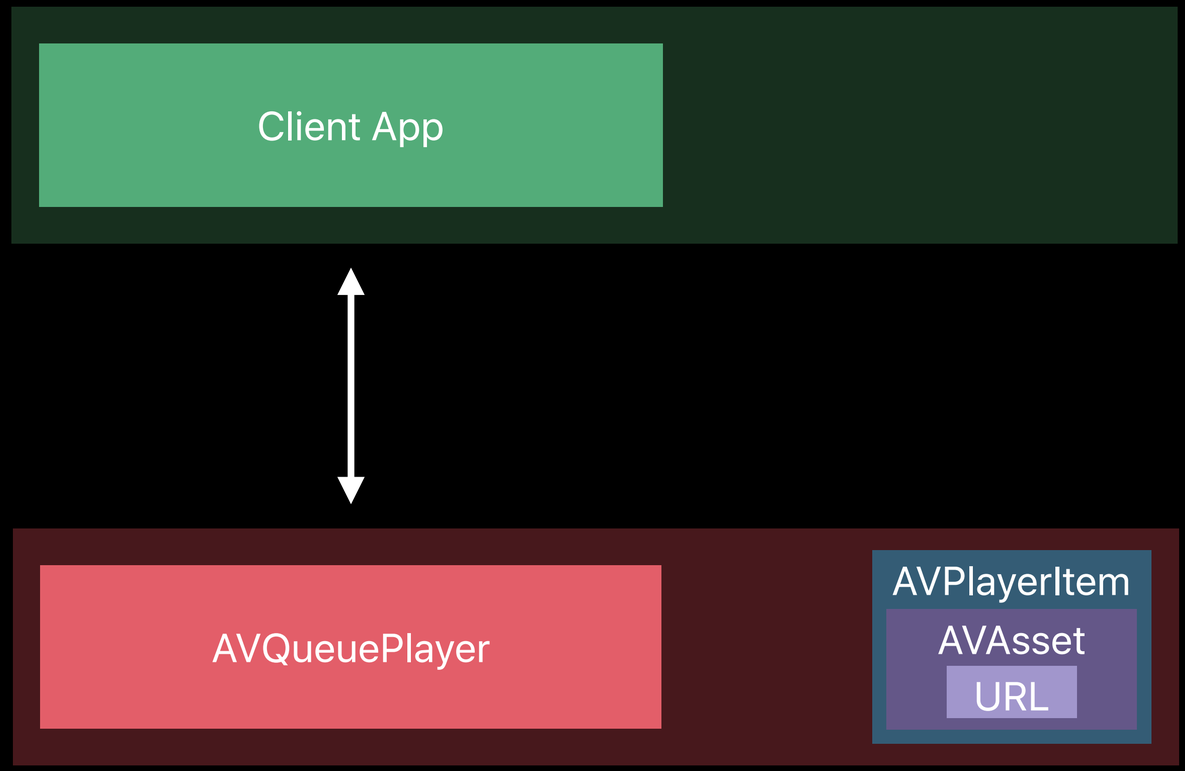
Enhanced Audio Buffering

AVPlayer / AVQueuePlayer



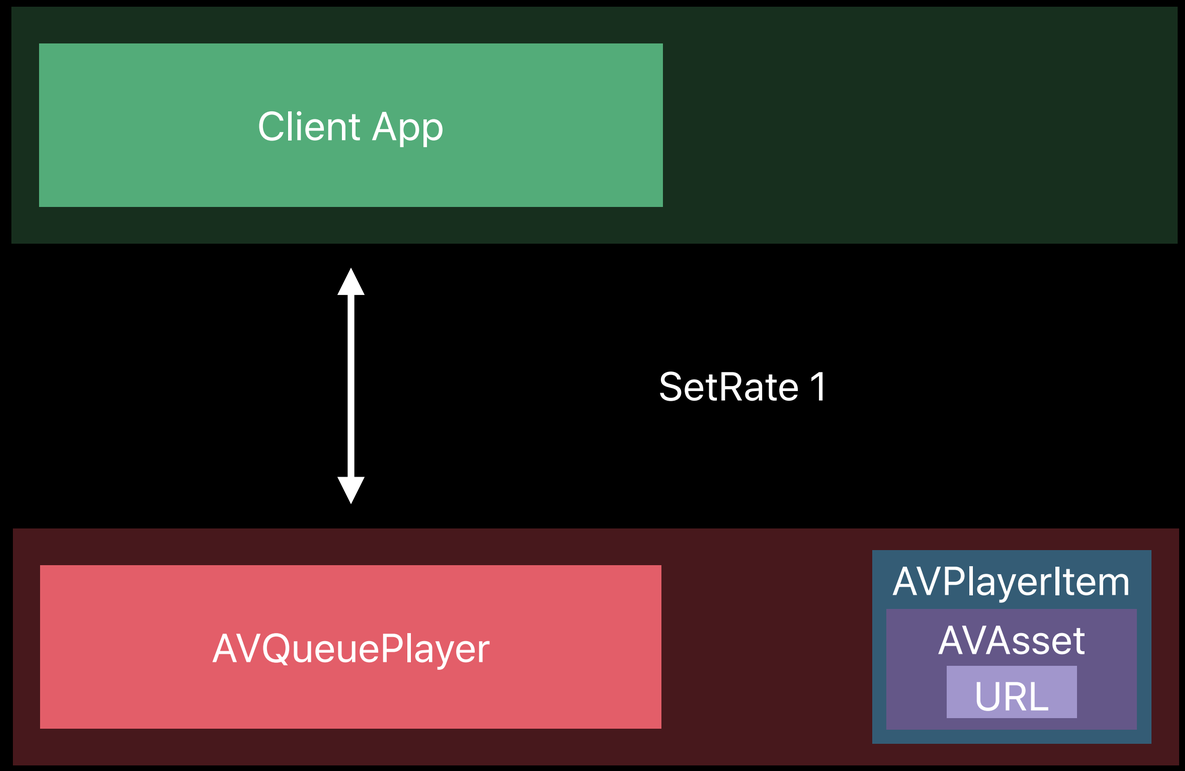
Enhanced Audio Buffering

AVPlayer / AVQueuePlayer



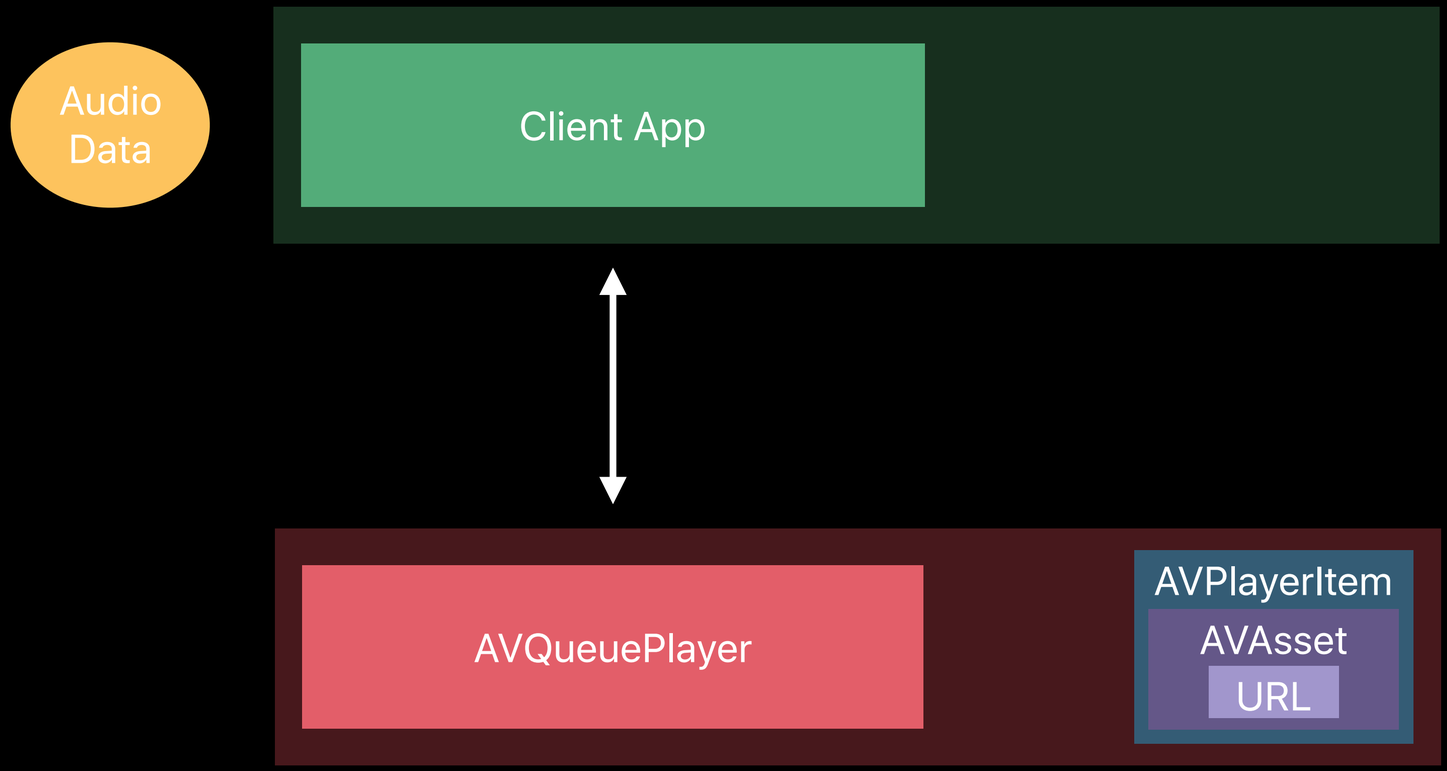
Enhanced Audio Buffering

AVPlayer / AVQueuePlayer



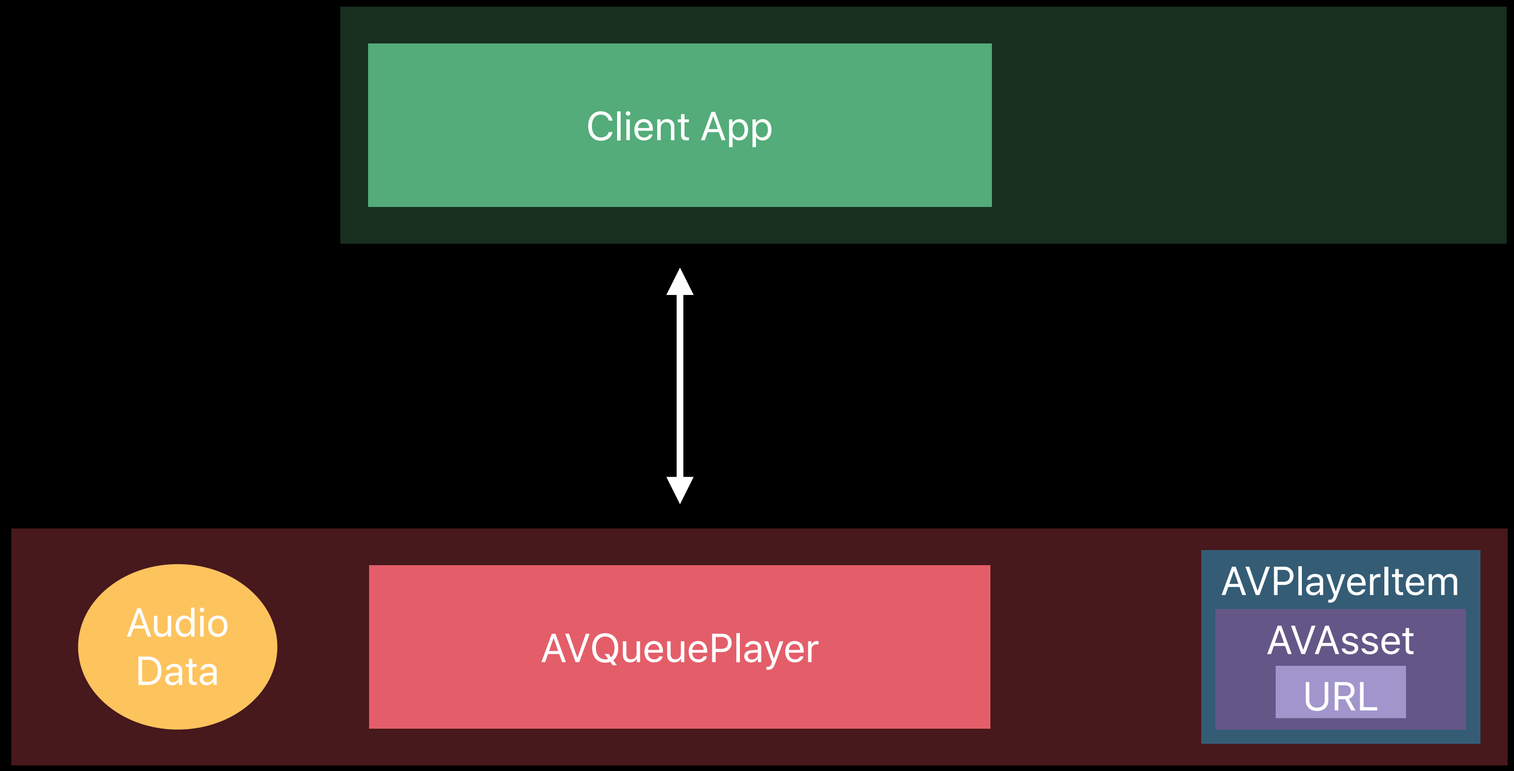
Enhanced Audio Buffering

AVPlayer / AVQueuePlayer



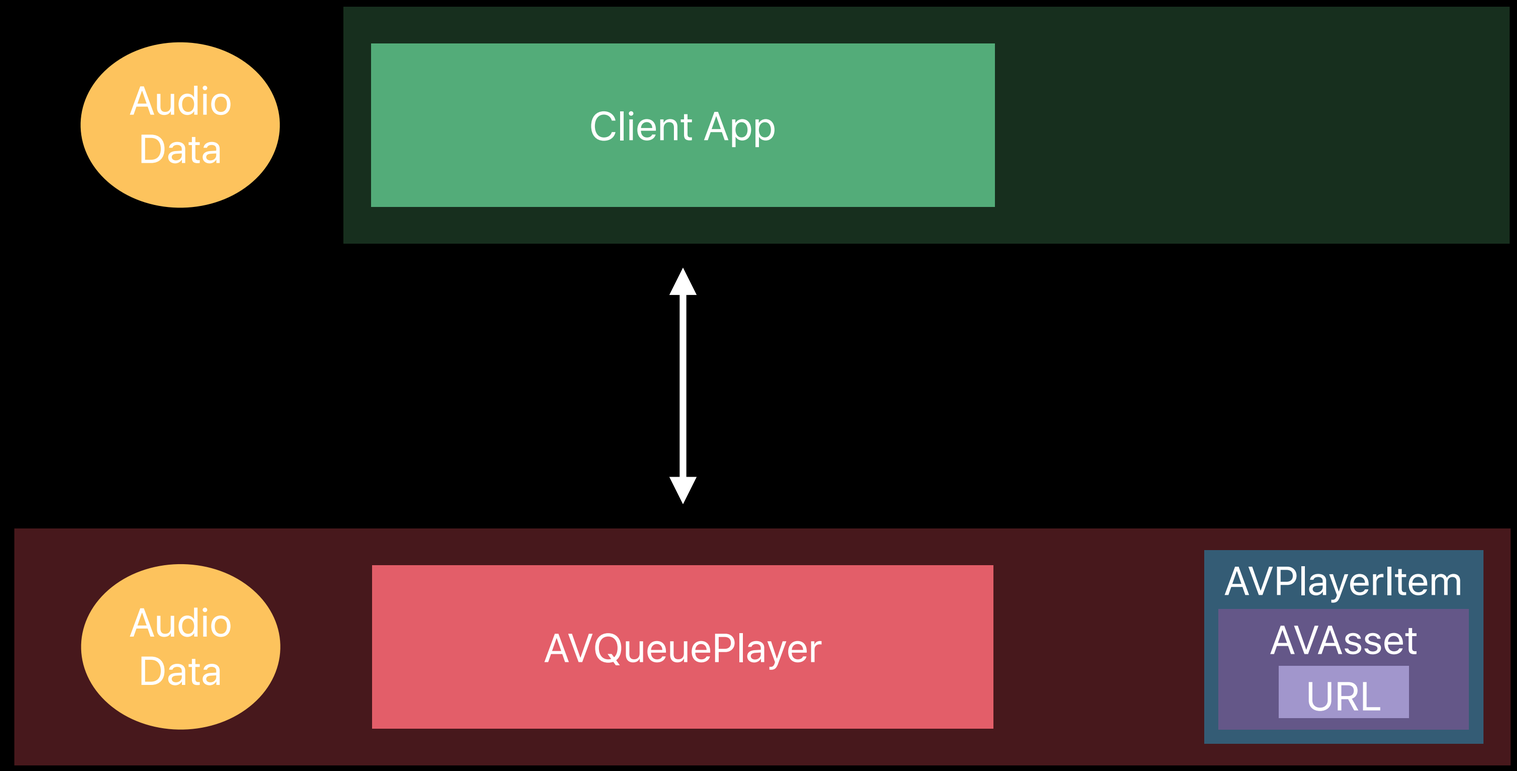
Enhanced Audio Buffering

AVPlayer / AVQueuePlayer



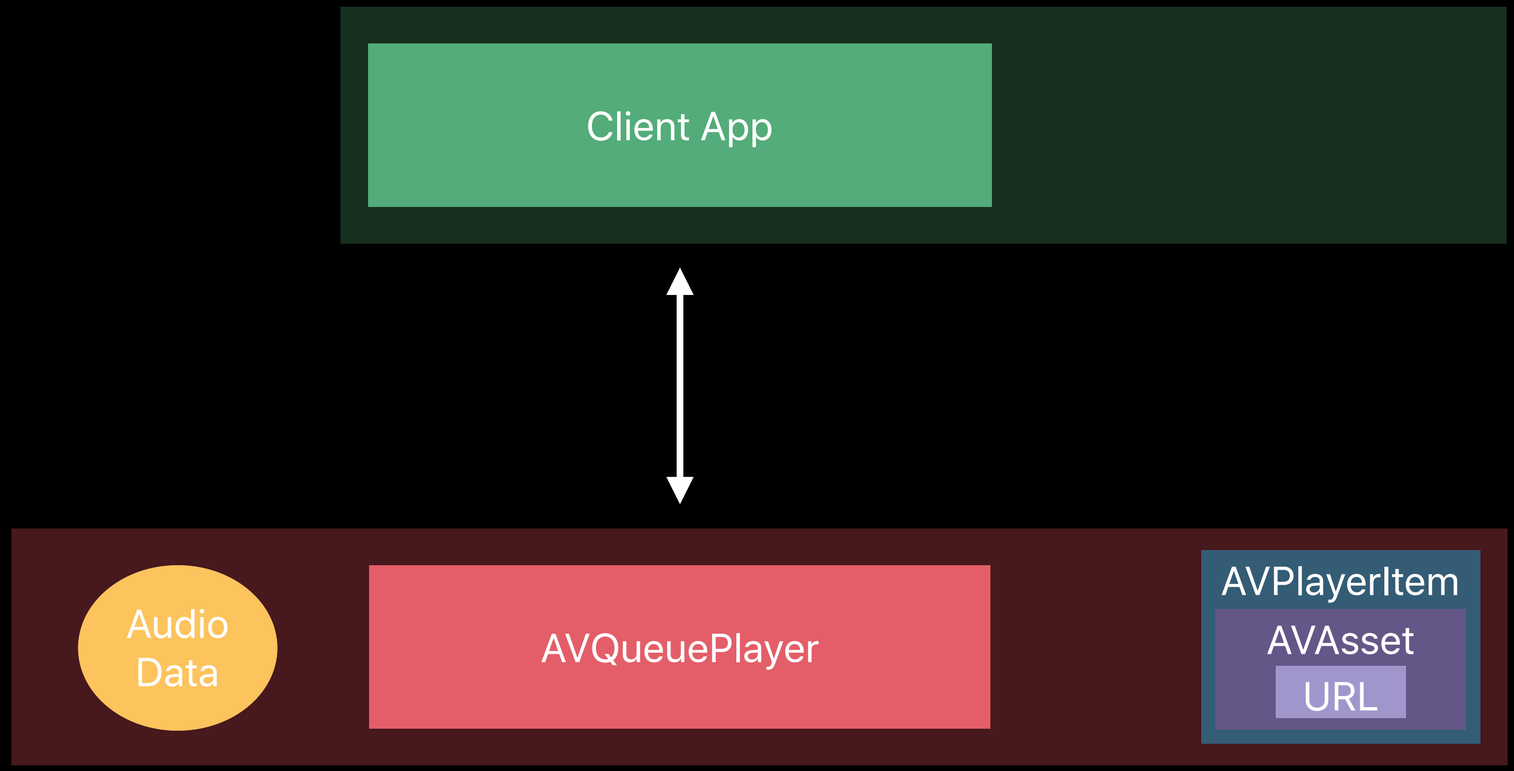
Enhanced Audio Buffering

AVPlayer / AVQueuePlayer



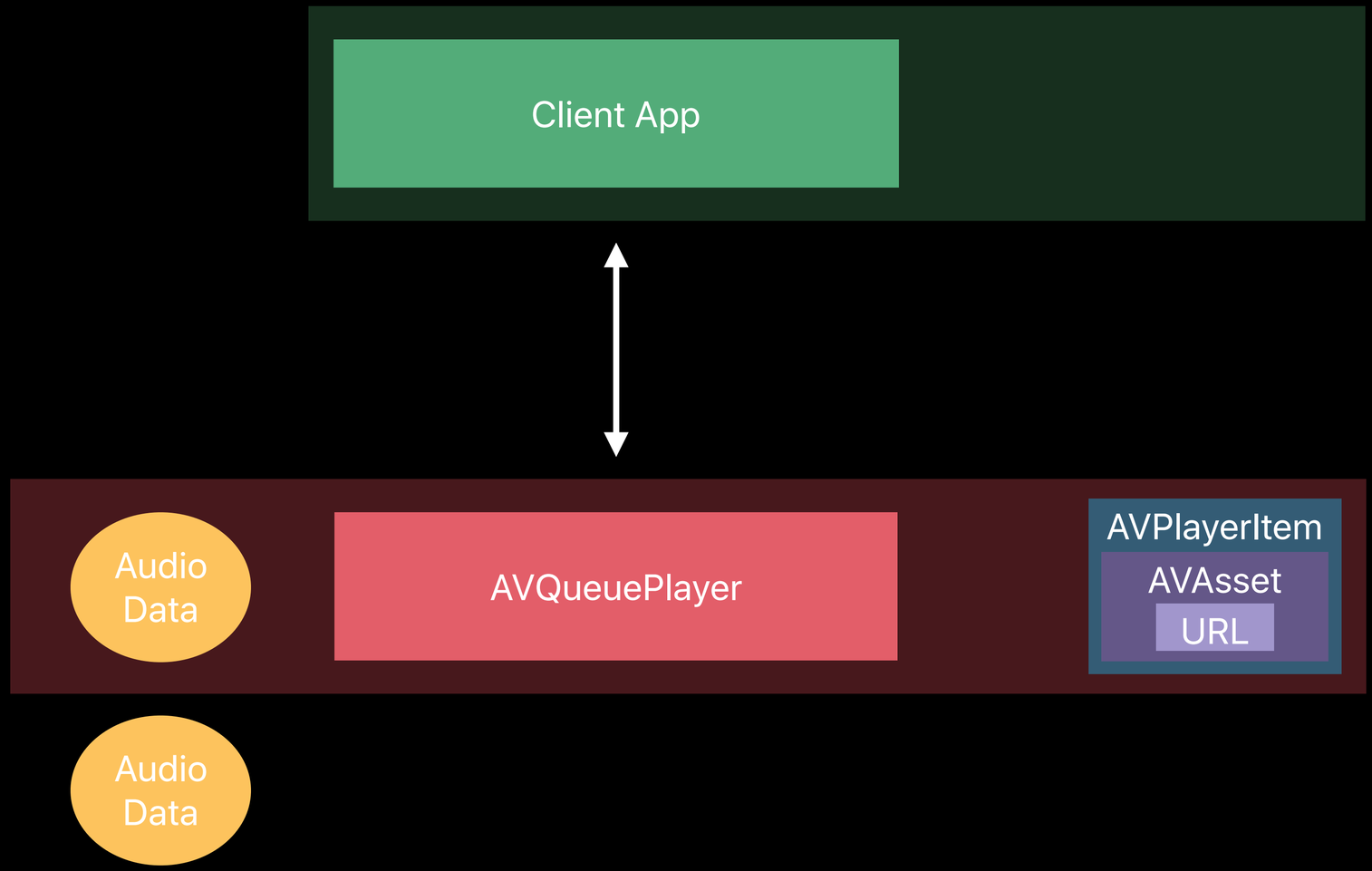
Enhanced Audio Buffering

AVPlayer / AVQueuePlayer



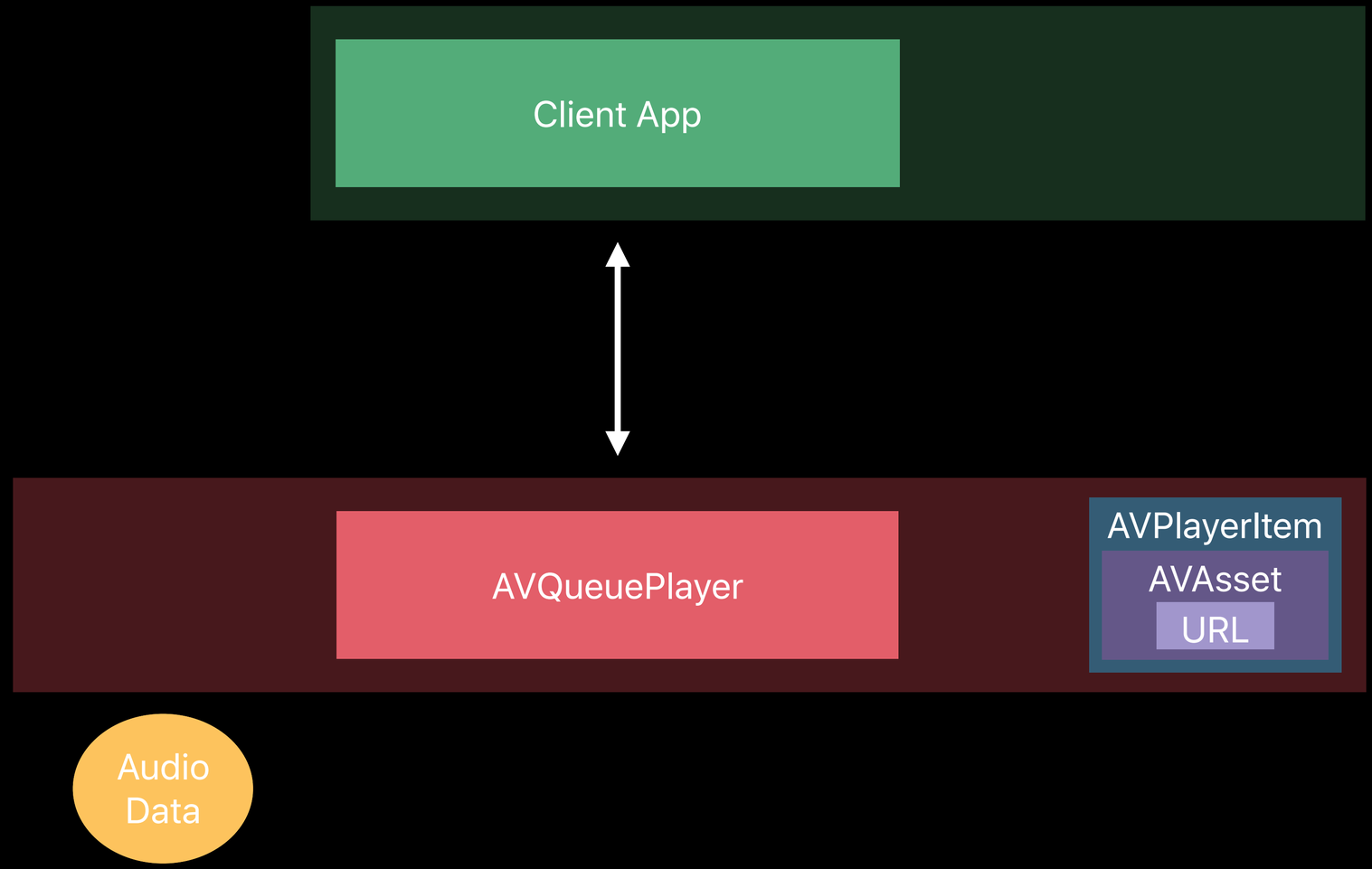
Enhanced Audio Buffering

AVPlayer / AVQueuePlayer



Enhanced Audio Buffering

AVPlayer / AVQueuePlayer



Enhanced Audio Buffering

AVSampleBufferAudioRenderer / AVSampleBufferRenderSynchronizer

NEW

Enhanced Audio Buffering

AVSampleBufferAudioRenderer / AVSampleBufferRenderSynchronizer

NEW

Your app has additional responsibilities

Enhanced Audio Buffering

AVSampleBufferAudioRenderer / AVSampleBufferRenderSynchronizer

NEW

Your app has additional responsibilities

- Sourcing and parsing the content

Enhanced Audio Buffering

AVSampleBufferAudioRenderer / AVSampleBufferRenderSynchronizer

NEW

Your app has additional responsibilities

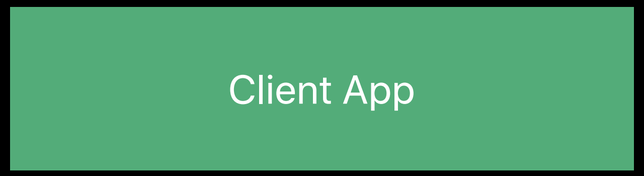
- Sourcing and parsing the content
- Providing raw audio buffers for rendering

Enhanced Audio Buffering

AVSampleBufferAudioRenderer / AVSampleBufferRenderSynchronizer

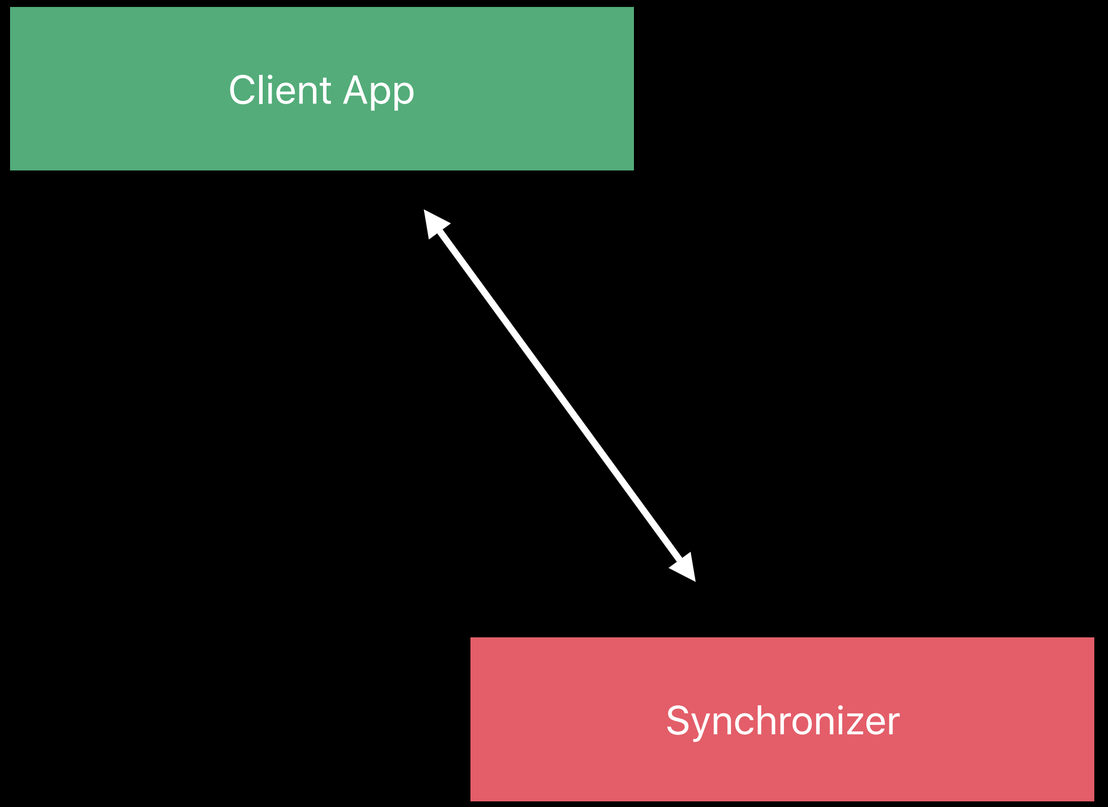
Enhanced Audio Buffering

AVSampleBufferAudioRenderer / AVSampleBufferRenderSynchronizer



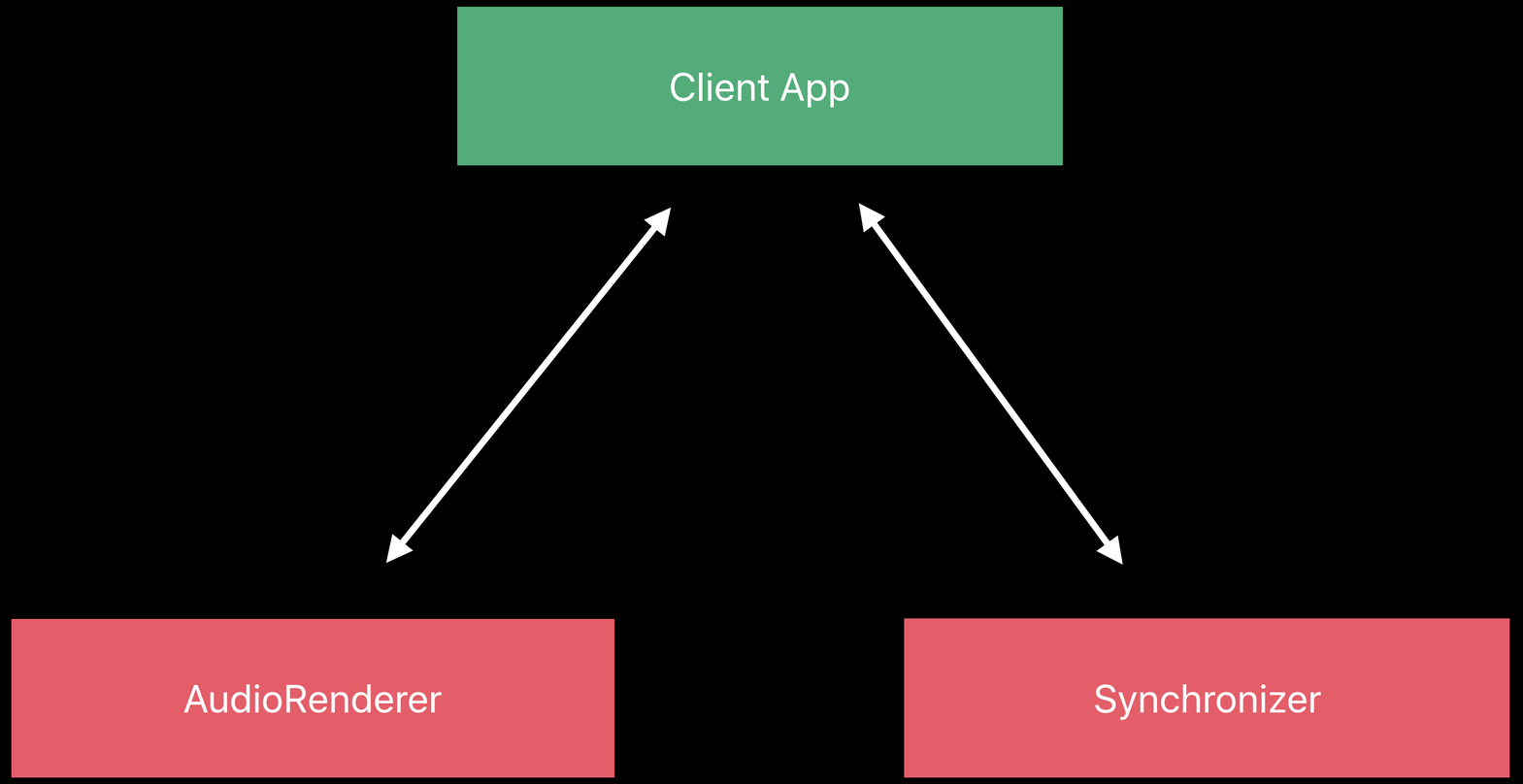
Enhanced Audio Buffering

AVSampleBufferAudioRenderer / AVSampleBufferRenderSynchronizer



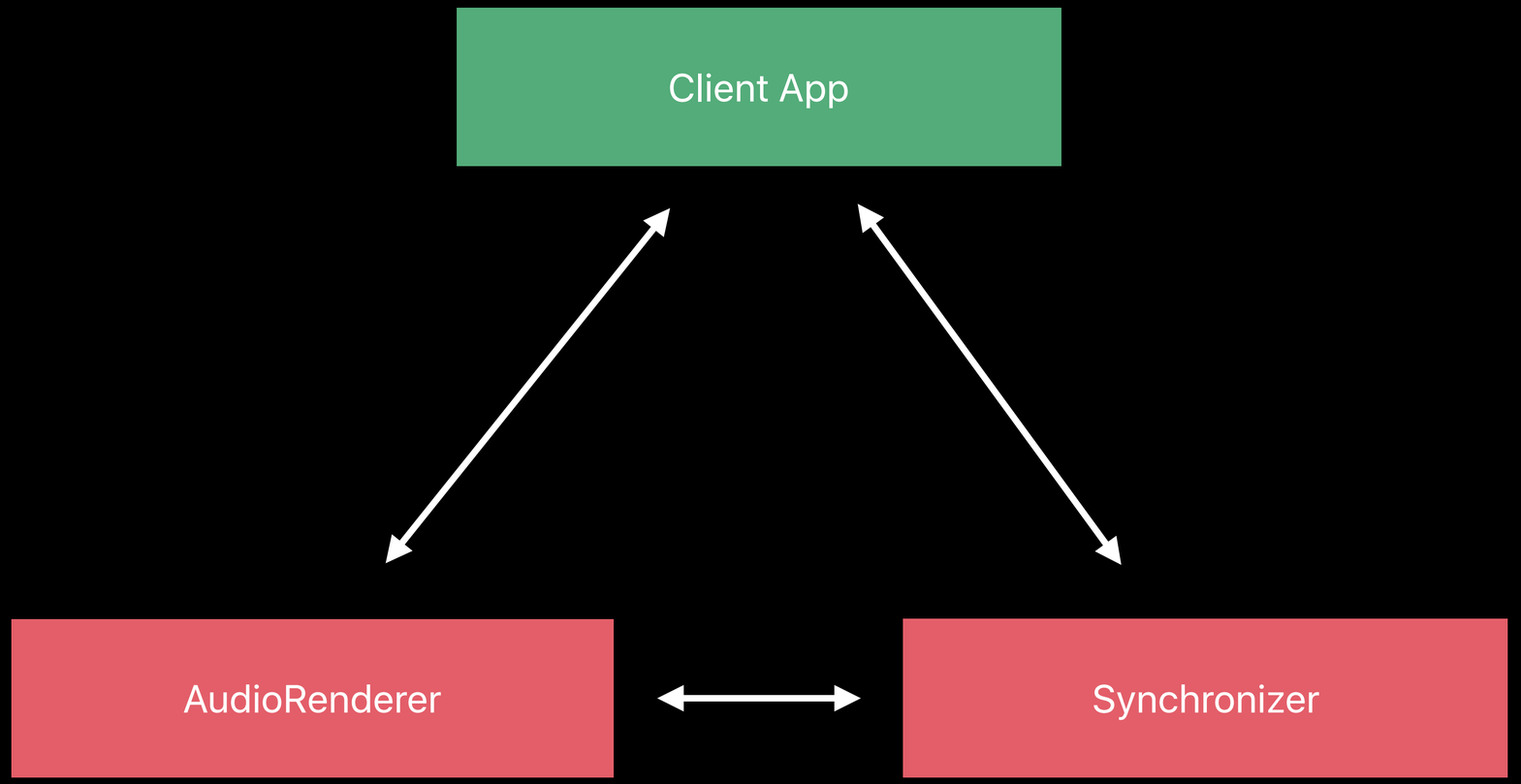
Enhanced Audio Buffering

AVSampleBufferAudioRenderer / AVSampleBufferRenderSynchronizer



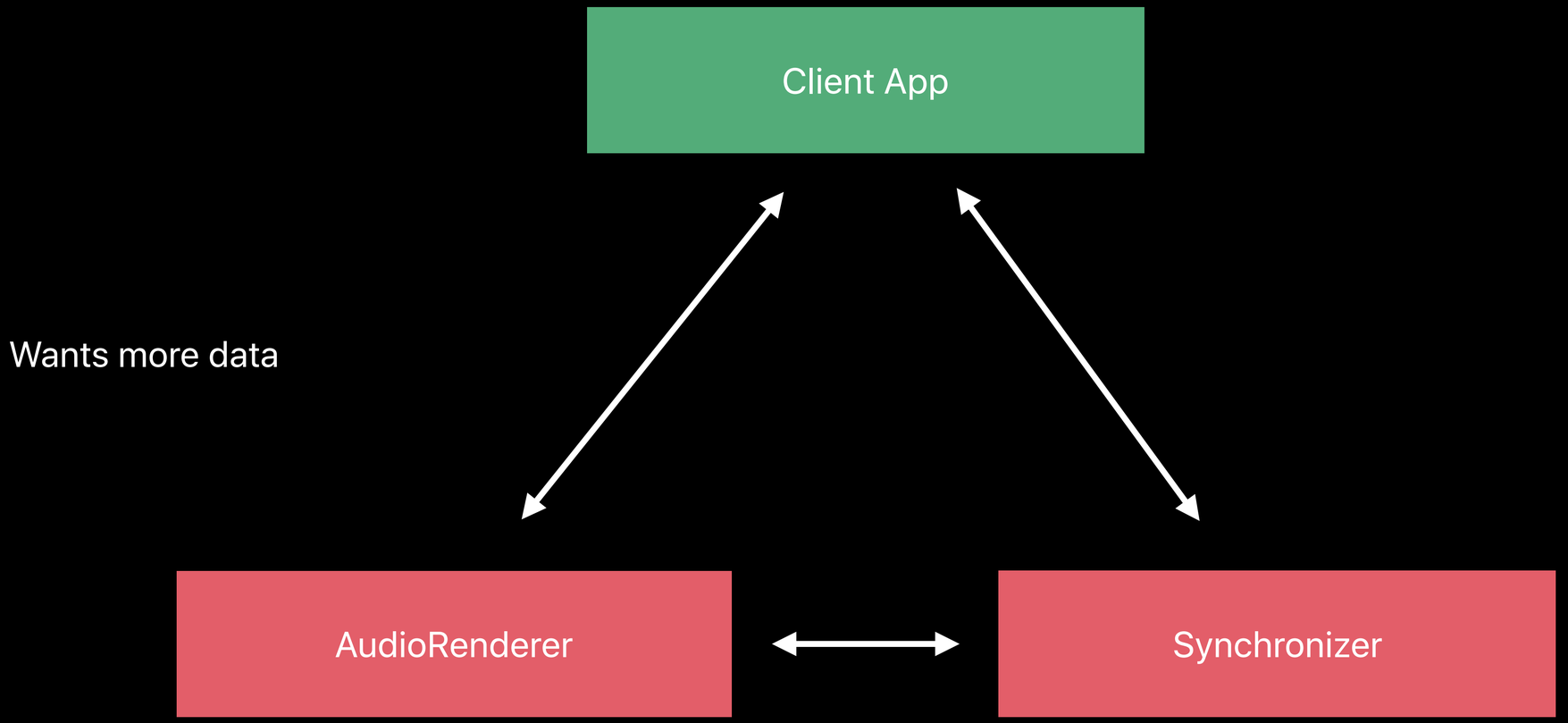
Enhanced Audio Buffering

AVSampleBufferAudioRenderer / AVSampleBufferRenderSynchronizer



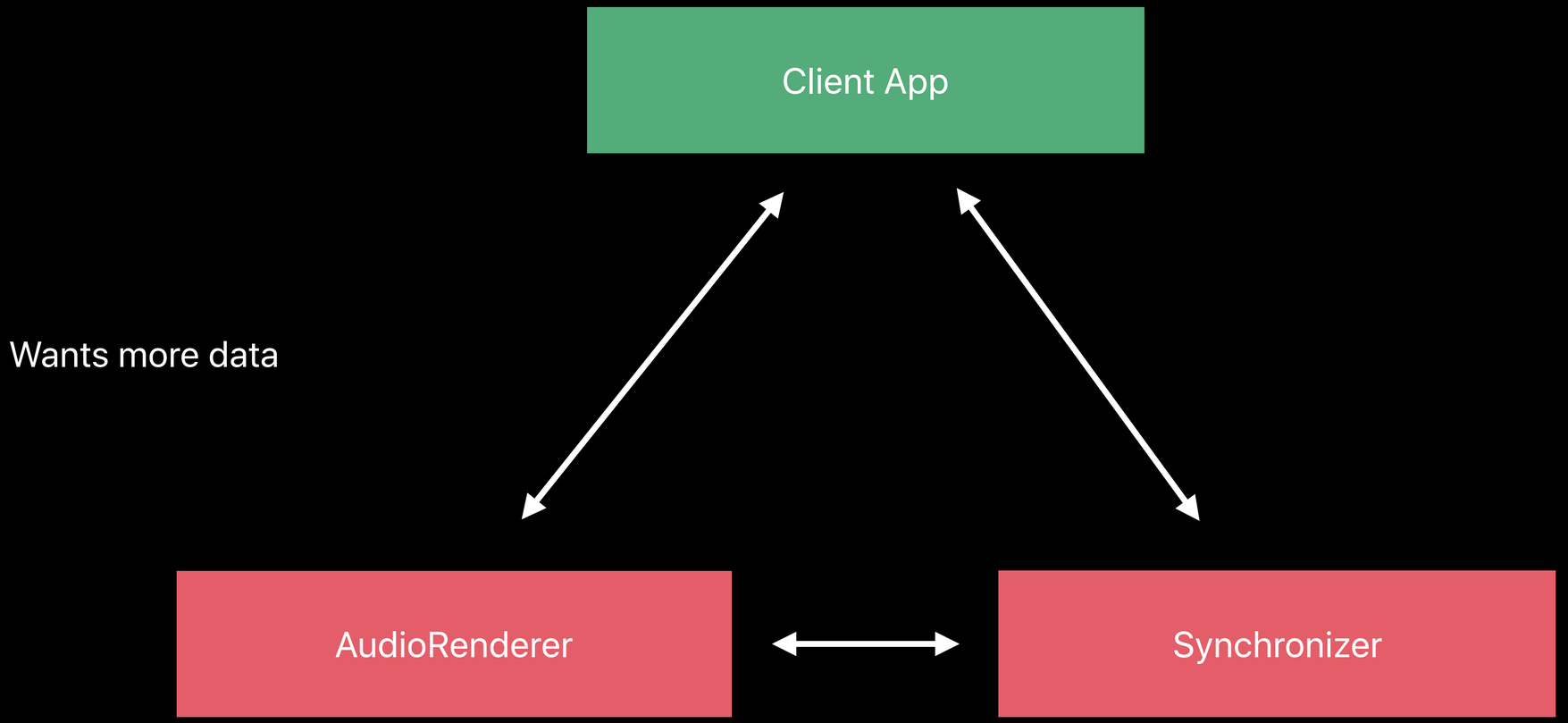
Enhanced Audio Buffering

AVSampleBufferAudioRenderer / AVSampleBufferRenderSynchronizer



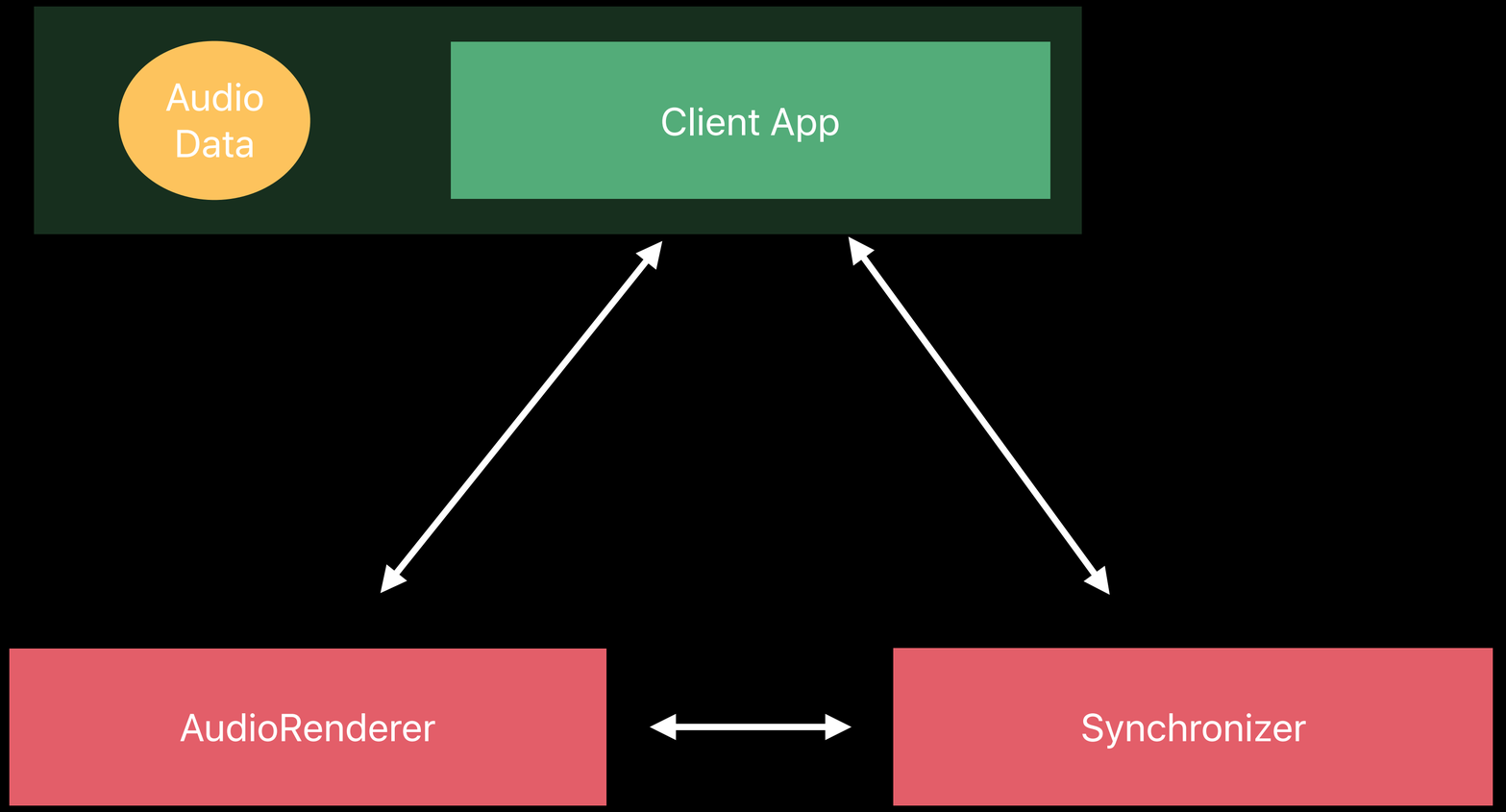
Enhanced Audio Buffering

AVSampleBufferAudioRenderer / AVSampleBufferRenderSynchronizer



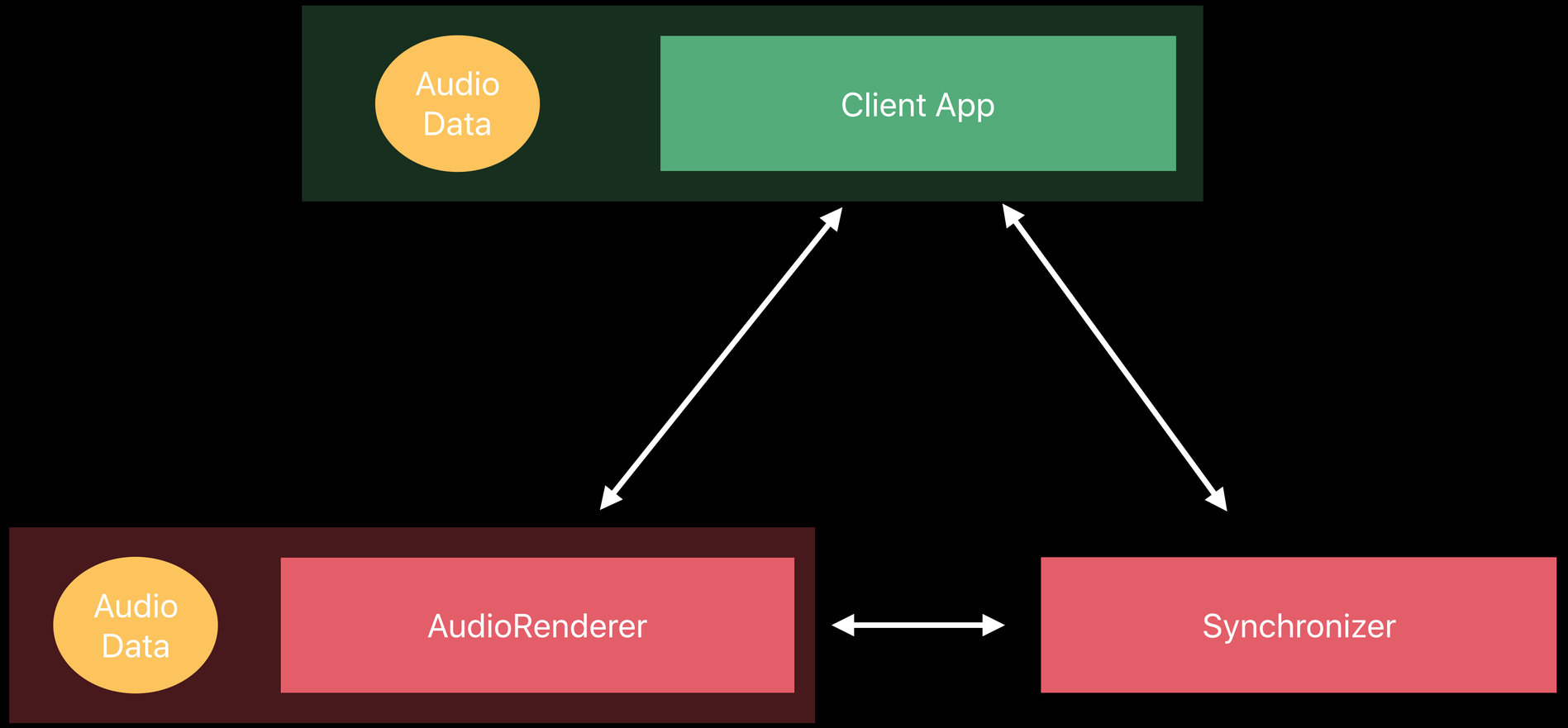
Enhanced Audio Buffering

AVSampleBufferAudioRenderer / AVSampleBufferRenderSynchronizer



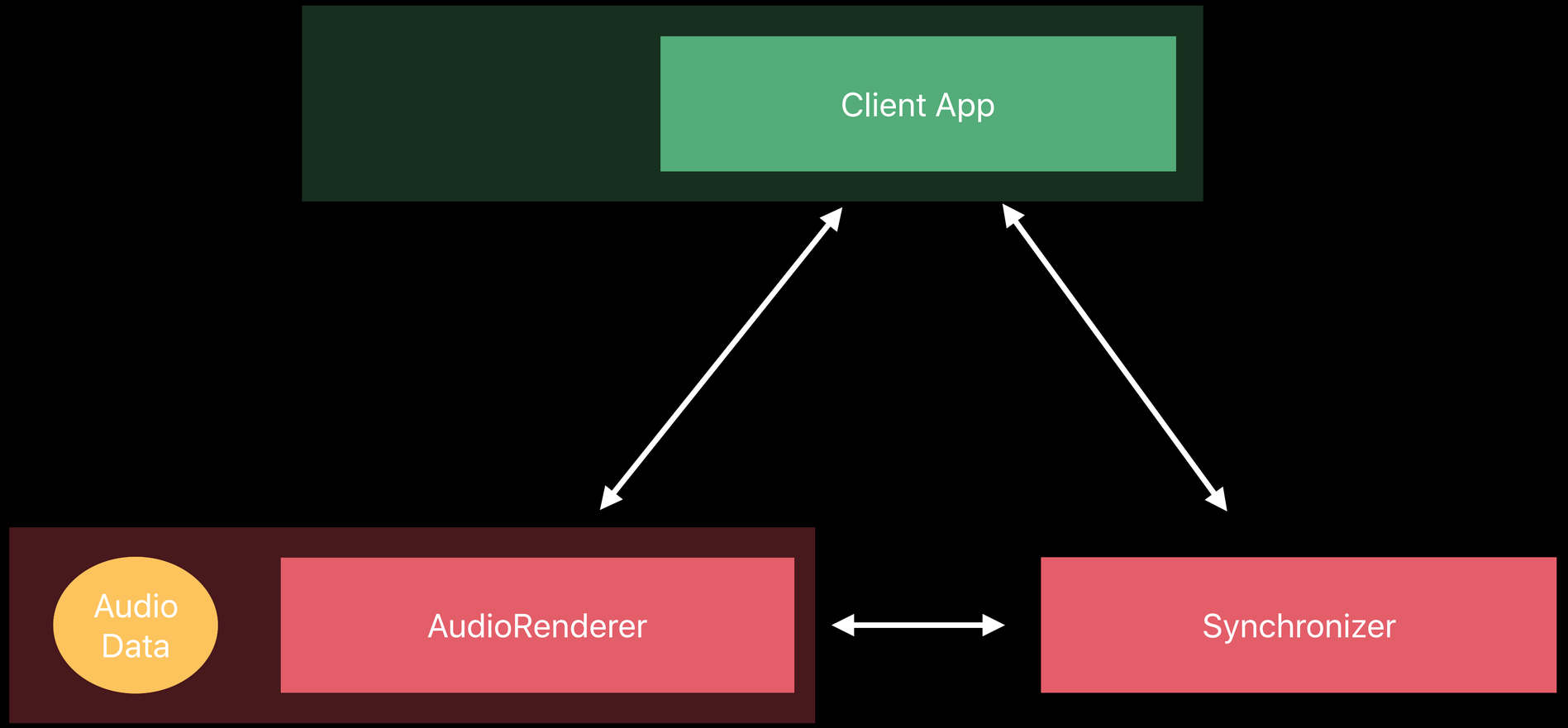
Enhanced Audio Buffering

AVSampleBufferAudioRenderer / AVSampleBufferRenderSynchronizer



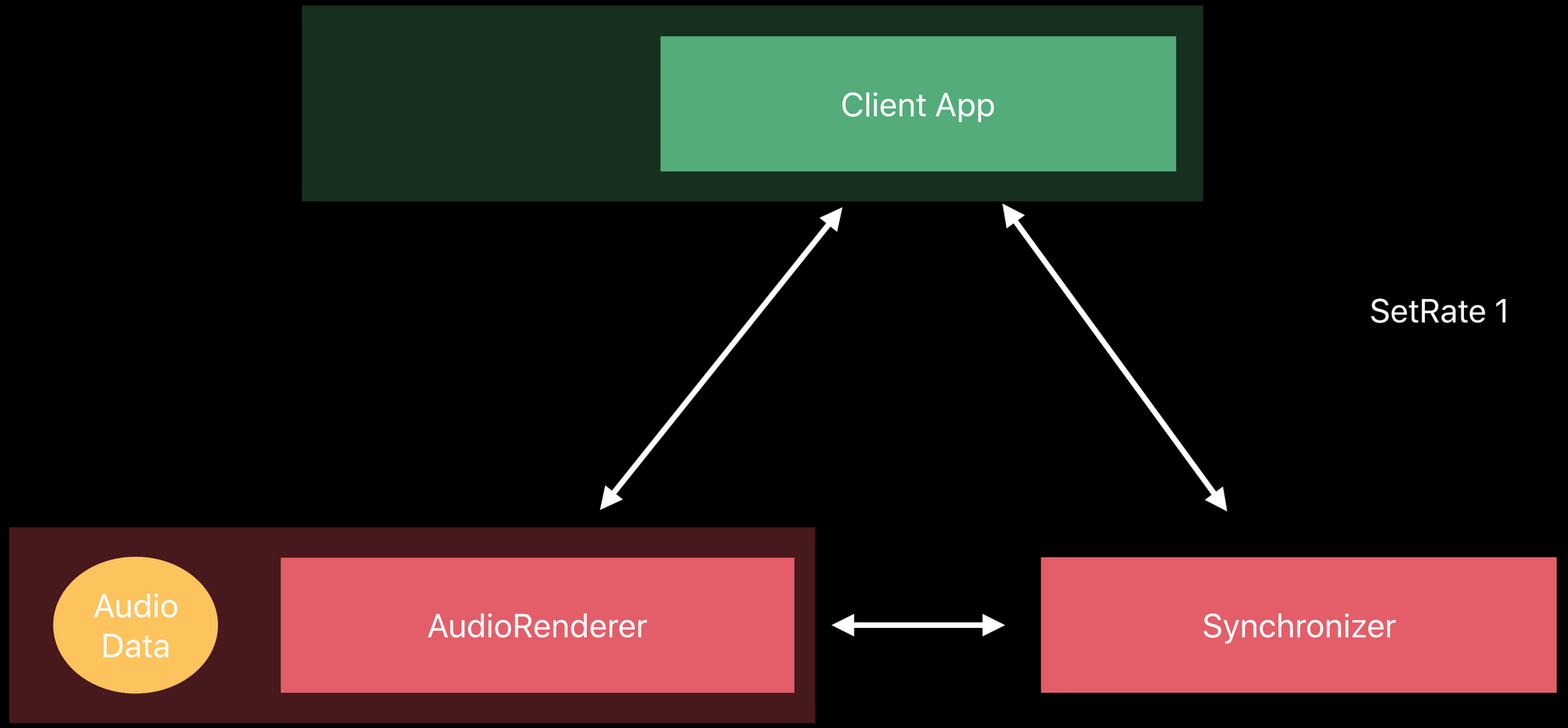
Enhanced Audio Buffering

AVSampleBufferAudioRenderer / AVSampleBufferRenderSynchronizer



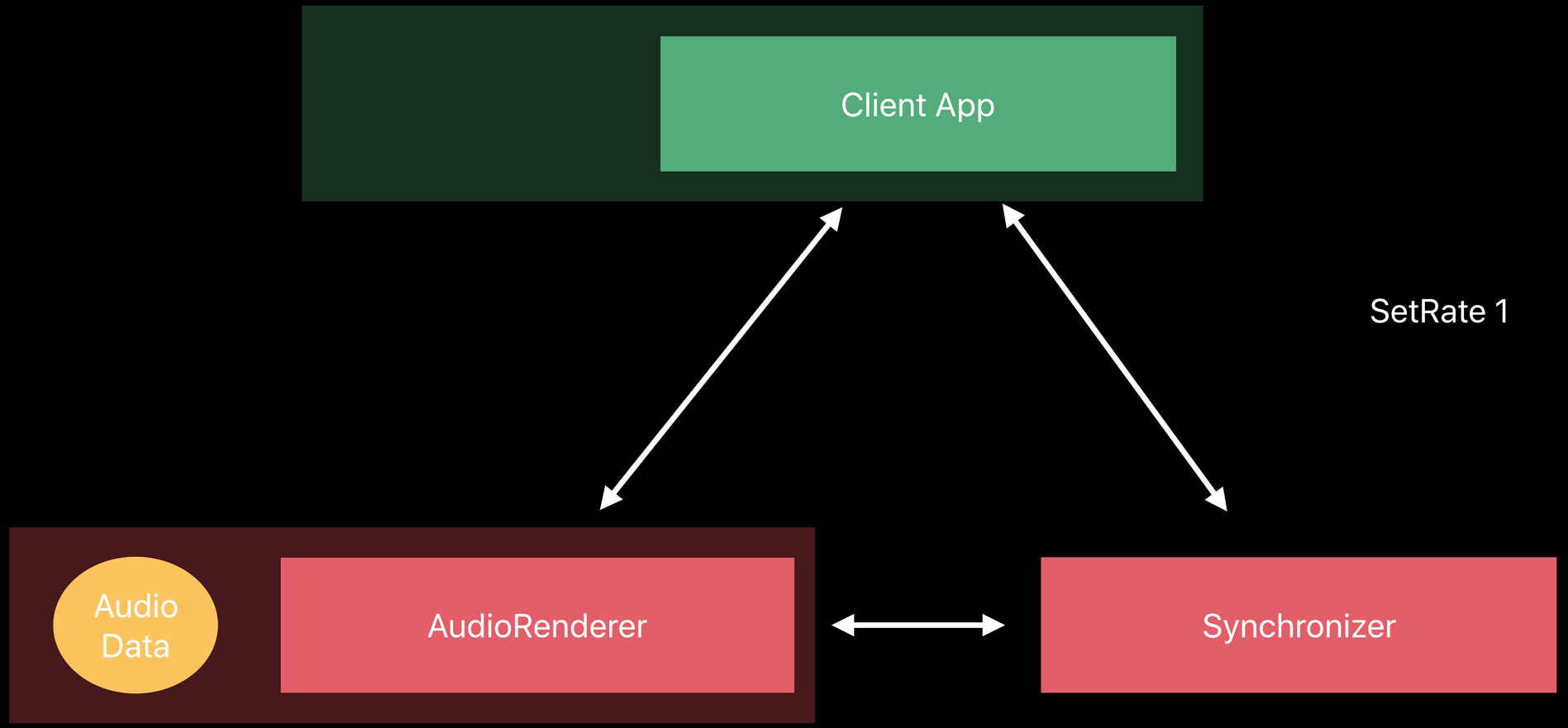
Enhanced Audio Buffering

AVSampleBufferAudioRenderer / AVSampleBufferRenderSynchronizer



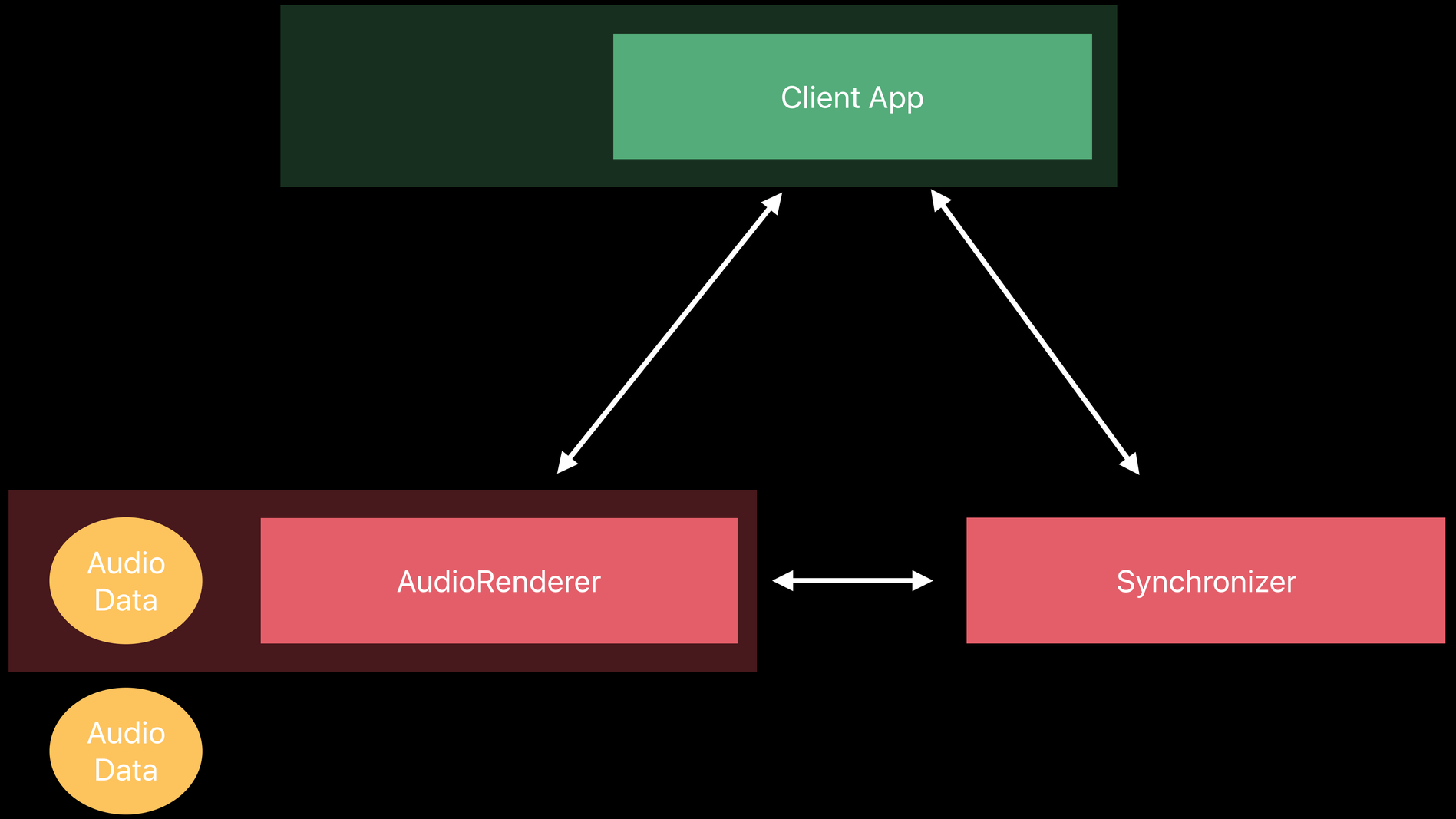
Enhanced Audio Buffering

AVSampleBufferAudioRenderer / AVSampleBufferRenderSynchronizer



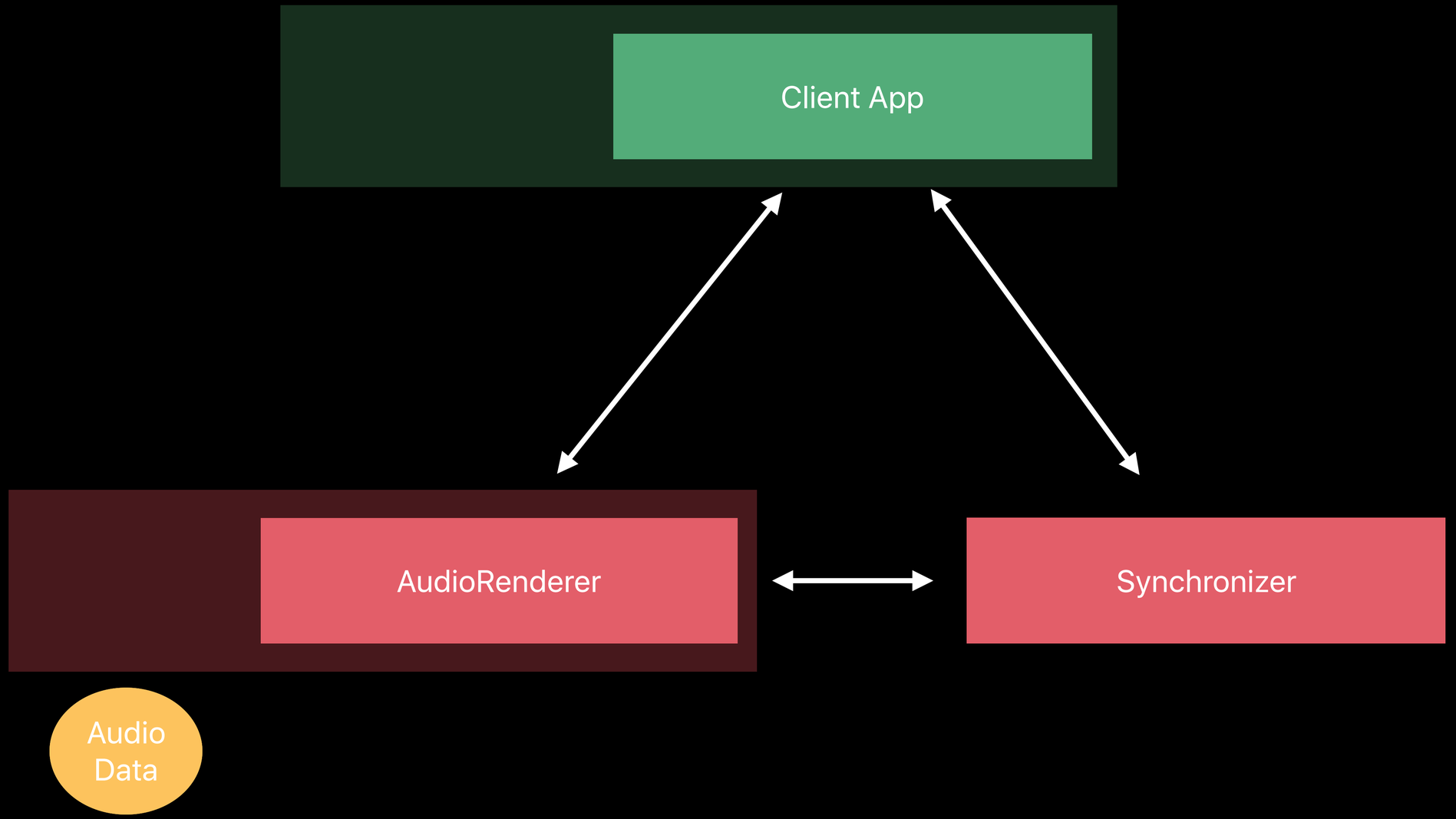
Enhanced Audio Buffering

AVSampleBufferAudioRenderer / AVSampleBufferRenderSynchronizer



Enhanced Audio Buffering

AVSampleBufferAudioRenderer / AVSampleBufferRenderSynchronizer



Demo

AirPlay 2 with enhanced buffering

Adam Sonnanstine, AVFoundation Engineer

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

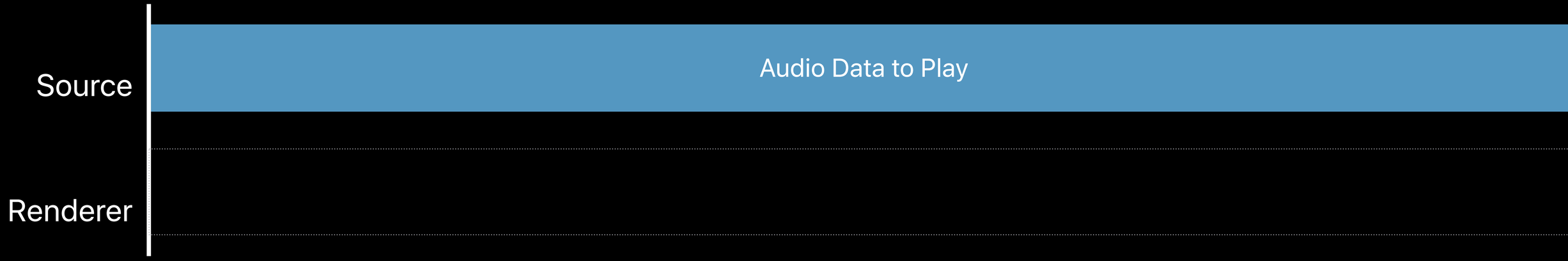
AVSampleBufferAudioRenderer

The amount of data requested will vary depending on the current route

Audio Buffer Levels

AVSampleBufferAudioRenderer

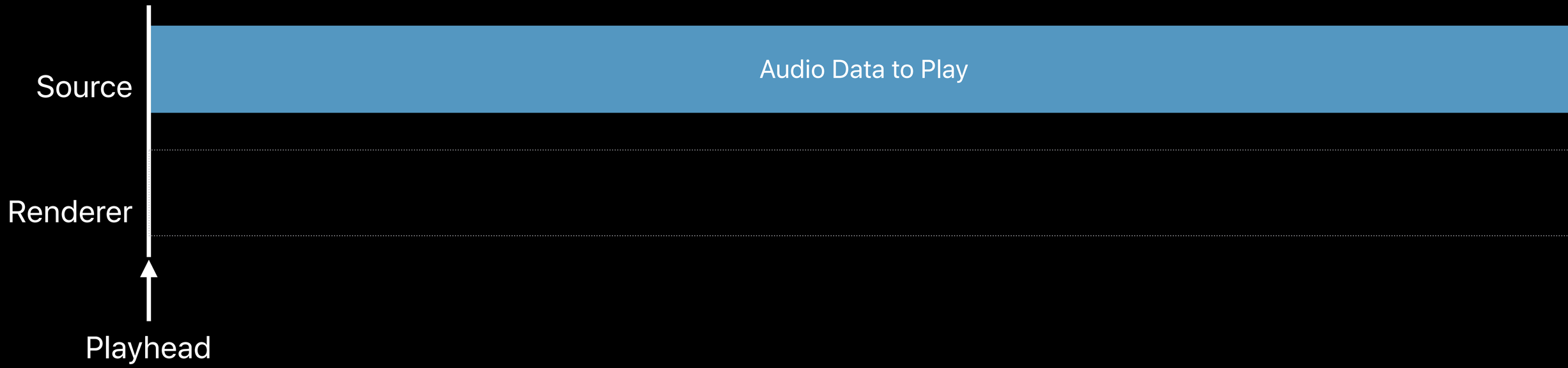
The amount of data requested will vary depending on the current route



Audio Buffer Levels

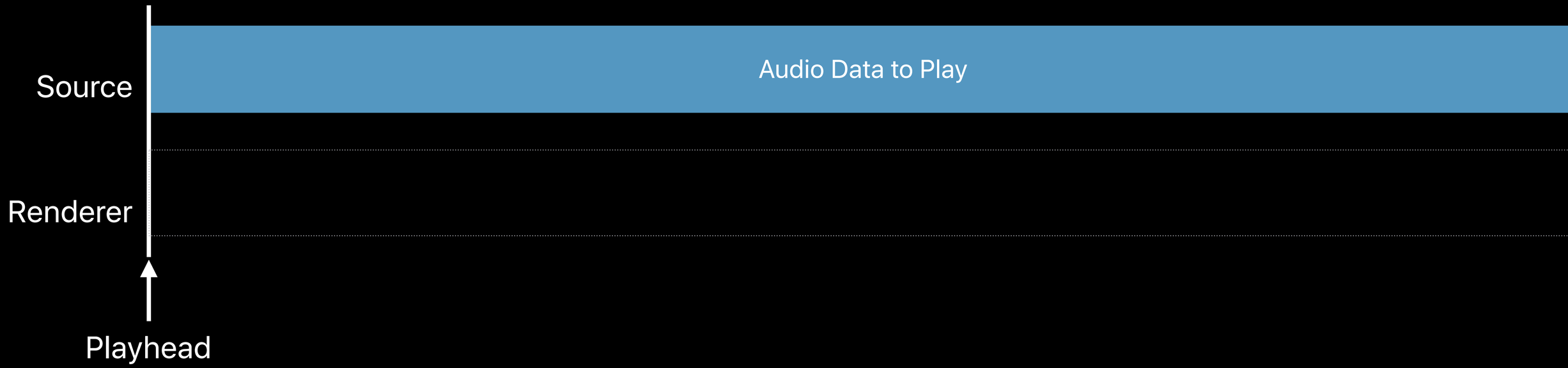
AVSampleBufferAudioRenderer

The amount of data requested will vary depending on the current route



Audio Buffer Levels

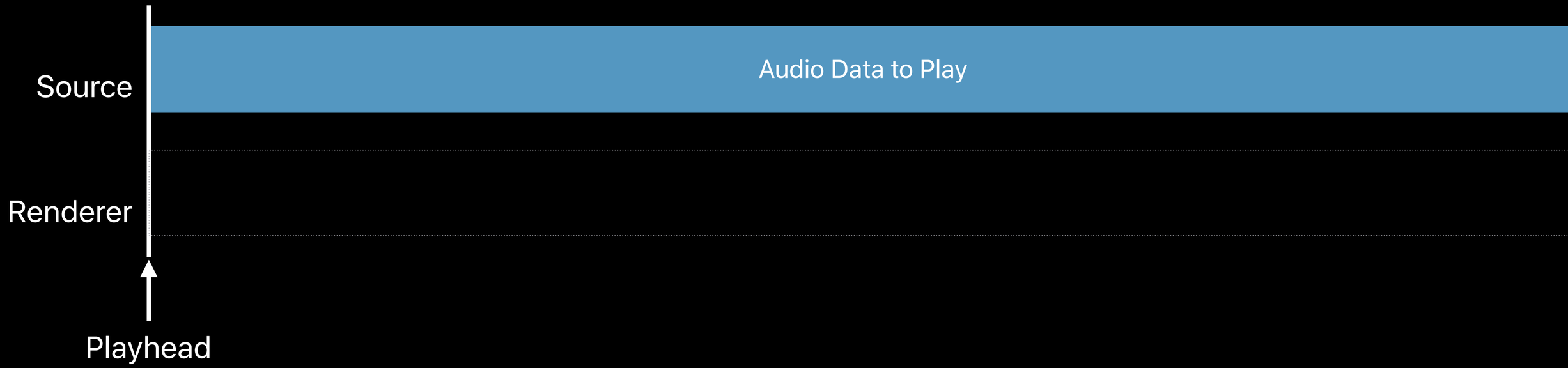
AVSampleBufferAudioRenderer



Audio Buffer Levels

AVSampleBufferAudioRenderer

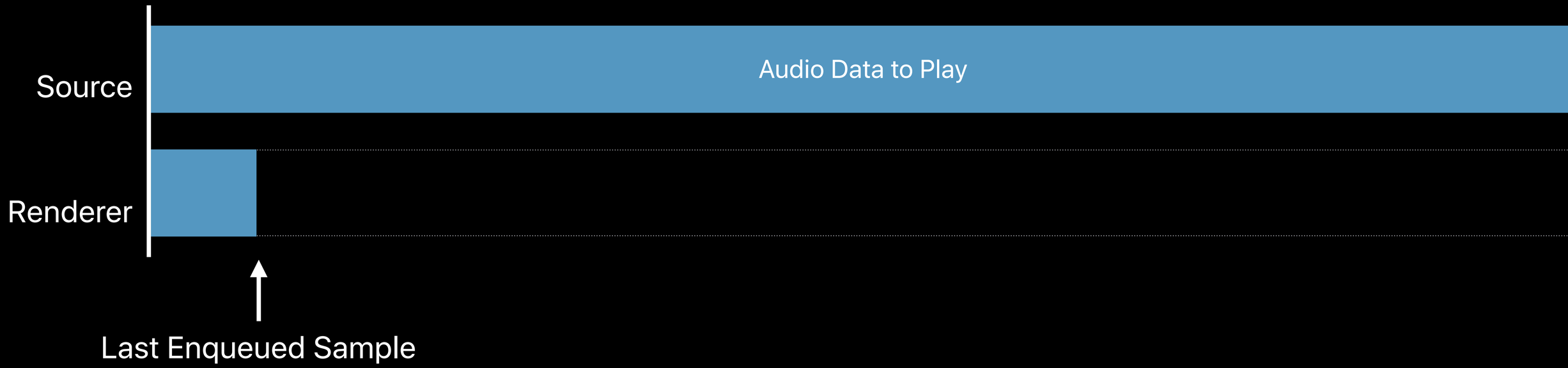
Local Playback: enqueue just a few seconds ahead of the Playhead



Audio Buffer Levels

AVSampleBufferAudioRenderer

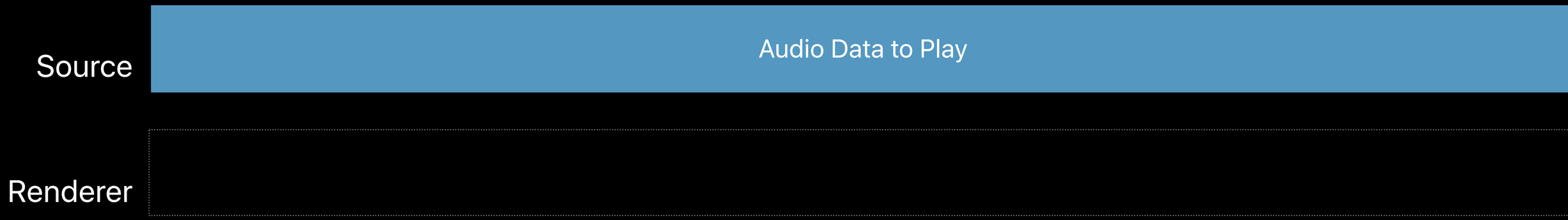
Local Playback: enqueue just a few seconds ahead of the Playhead



Audio Buffer Levels

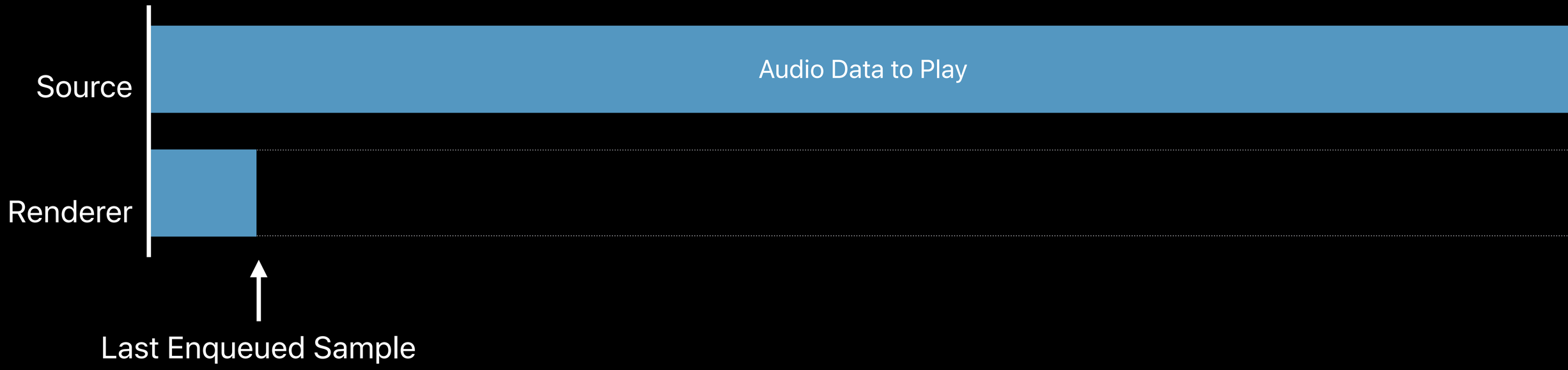
AVSampleBufferAudioRenderer

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



Audio Buffer Levels

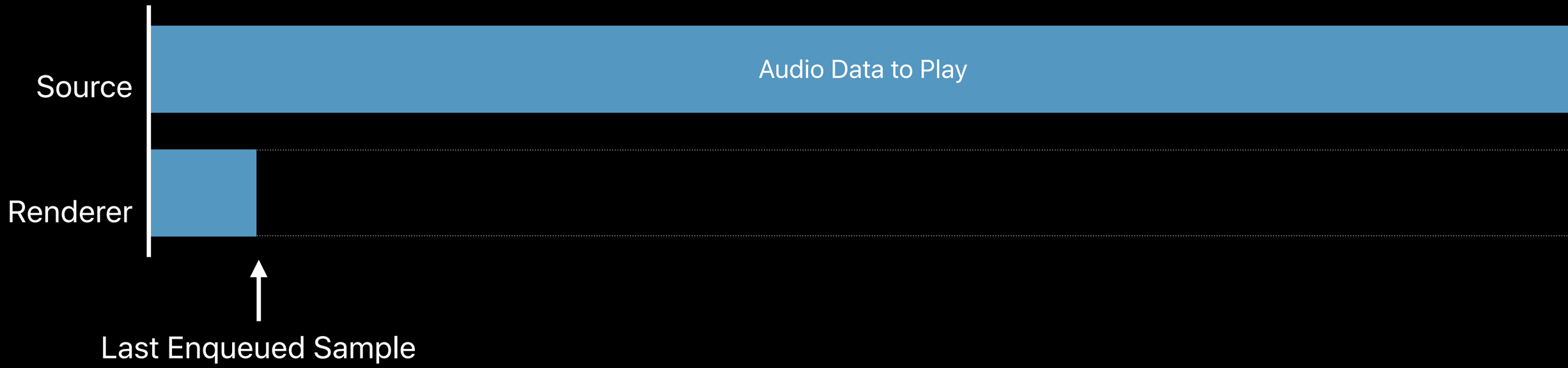
AVSampleBufferAudioRenderer



Audio Buffer Levels

AVSampleBufferAudioRenderer

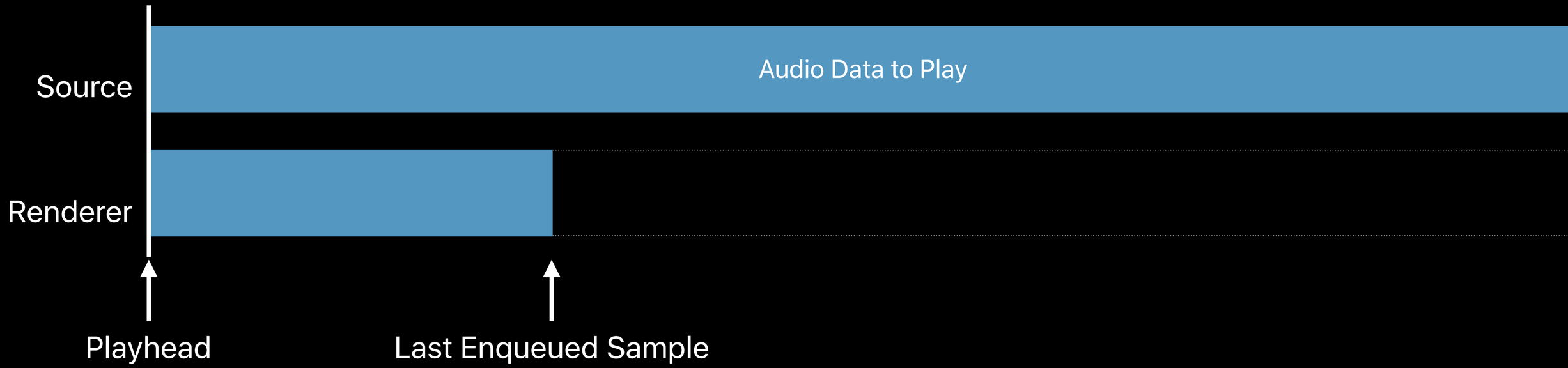
AirPlay 2 Speaker: enqueue up to multiple minutes of the Playhead



Audio Buffer Levels

AVSampleBufferAudioRenderer

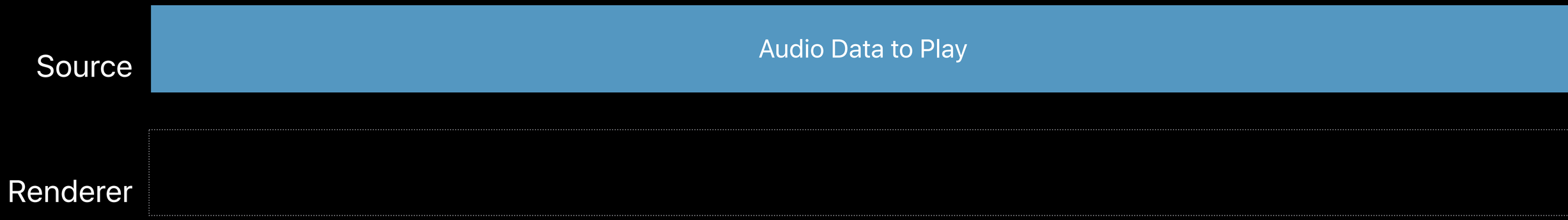
AirPlay 2 Speaker: enqueue up to multiple minutes of the Playhead



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

Audio Buffer Levels

AVSampleBufferAudioRenderer

Requested data amount varies by audio route

Local	Seconds

Audio Buffer Levels

AVSampleBufferAudioRenderer

Requested data amount varies by audio route

Local	Seconds
Bluetooth	Seconds

Audio Buffer Levels

AVSampleBufferAudioRenderer

Requested data amount varies by audio route

Local	Seconds
Bluetooth	Seconds
AirPlay	Seconds

Audio Buffer Levels

AVSampleBufferAudioRenderer

Requested data amount varies by audio route

Local	Seconds
Bluetooth	Seconds
AirPlay	Seconds
AirPlay 2	Minutes

Audio Buffer Levels

AVSampleBufferAudioRenderer

Requested data amount varies by audio route

Local	Seconds
Bluetooth	Seconds
AirPlay	Seconds
AirPlay 2	Minutes

Your app should handle these request changes

Seek

AVSampleBufferAudioRenderer

Seek

AVSampleBufferAudioRenderer

Seek

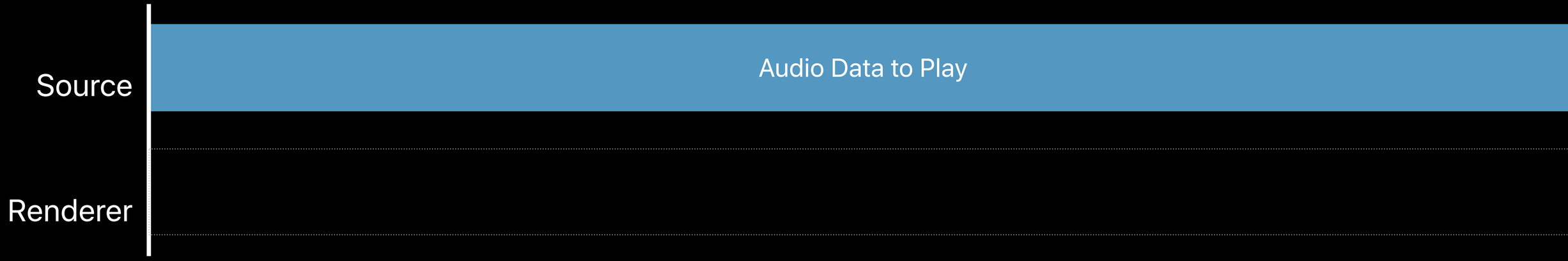
AVSampleBufferAudioRenderer

Manually changing Playhead location

Seek

AVSampleBufferAudioRenderer

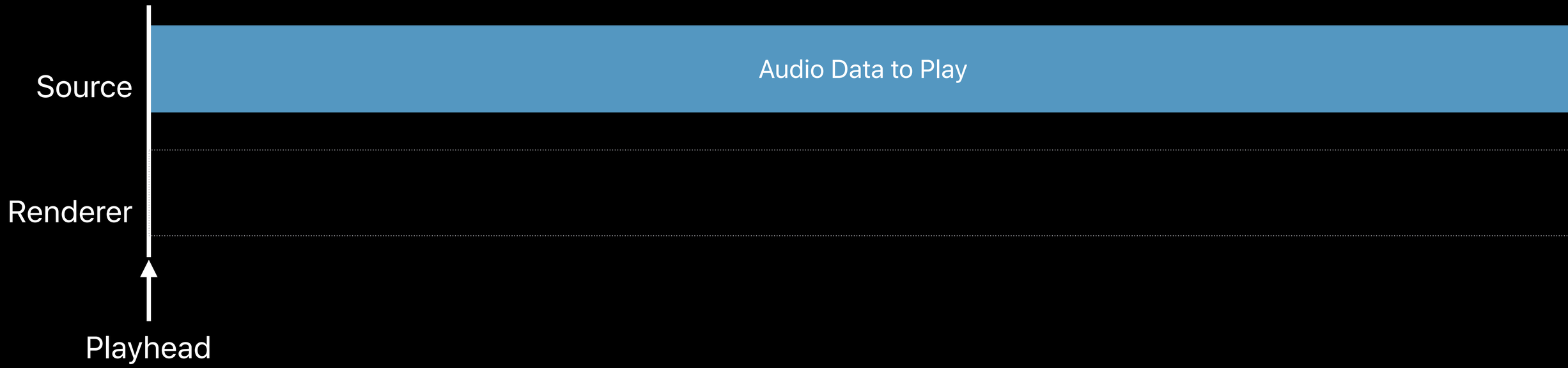
Manually changing Playhead location



Seek

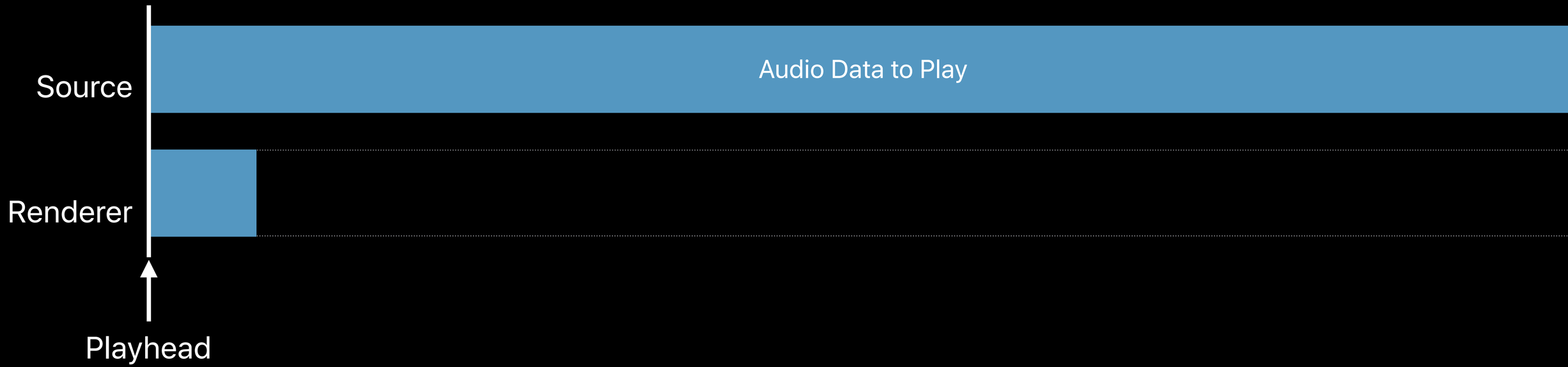
AVSampleBufferAudioRenderer

Manually changing Playhead location



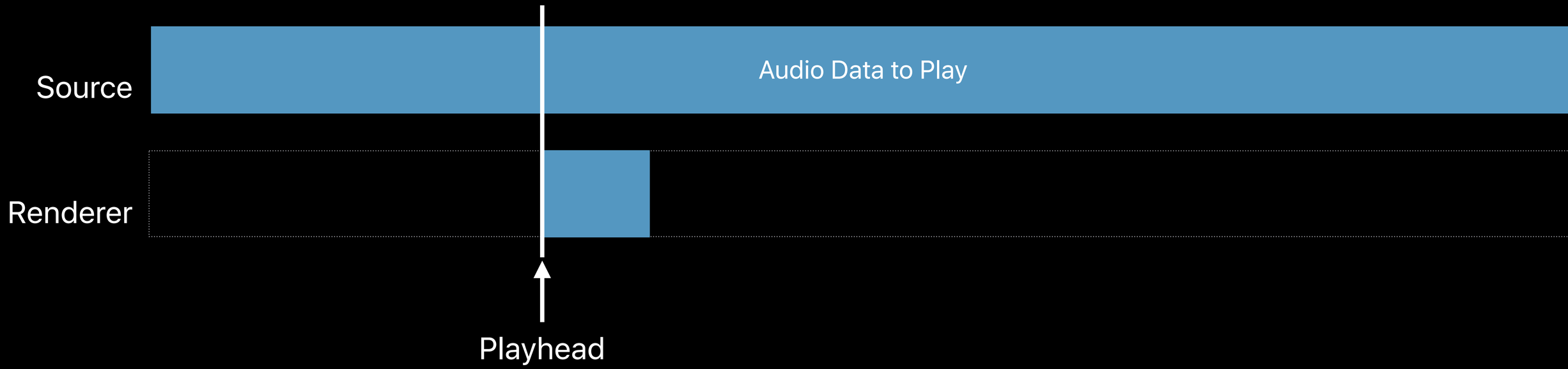
Seek

AVSampleBufferAudioRenderer



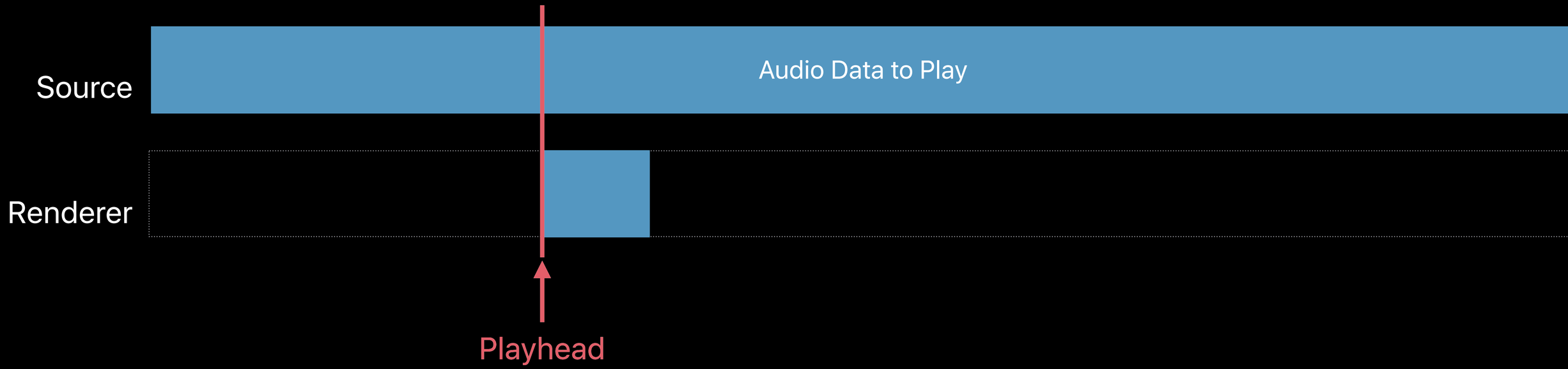
Seek

AVSampleBufferAudioRenderer



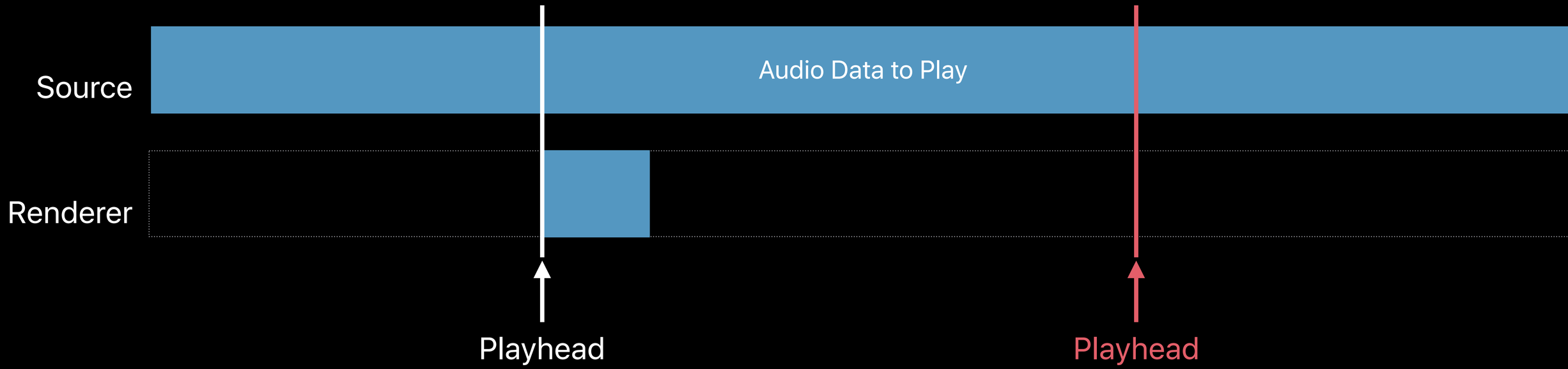
Seek

AVSampleBufferAudioRenderer



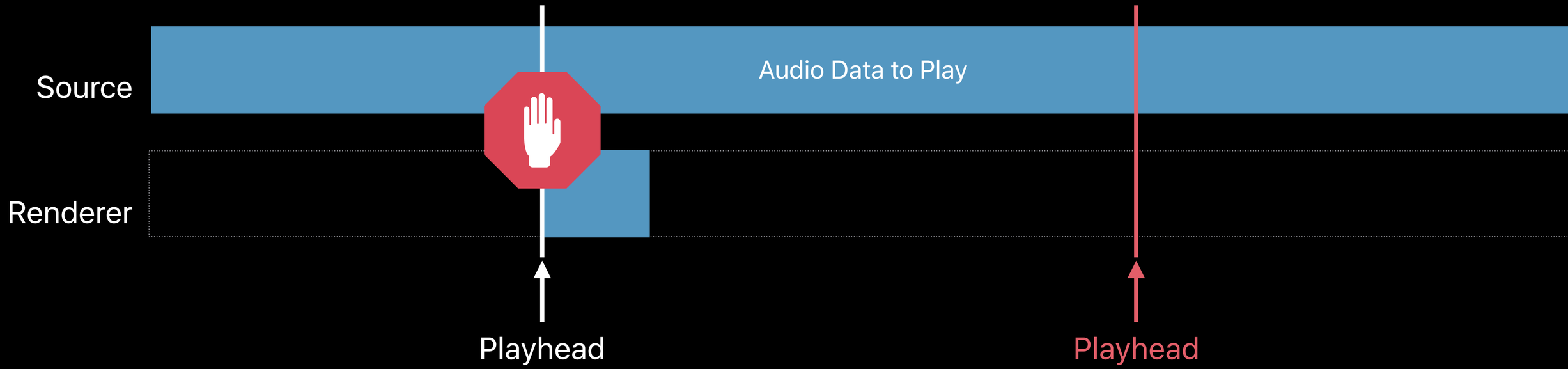
Seek

AVSampleBufferAudioRenderer



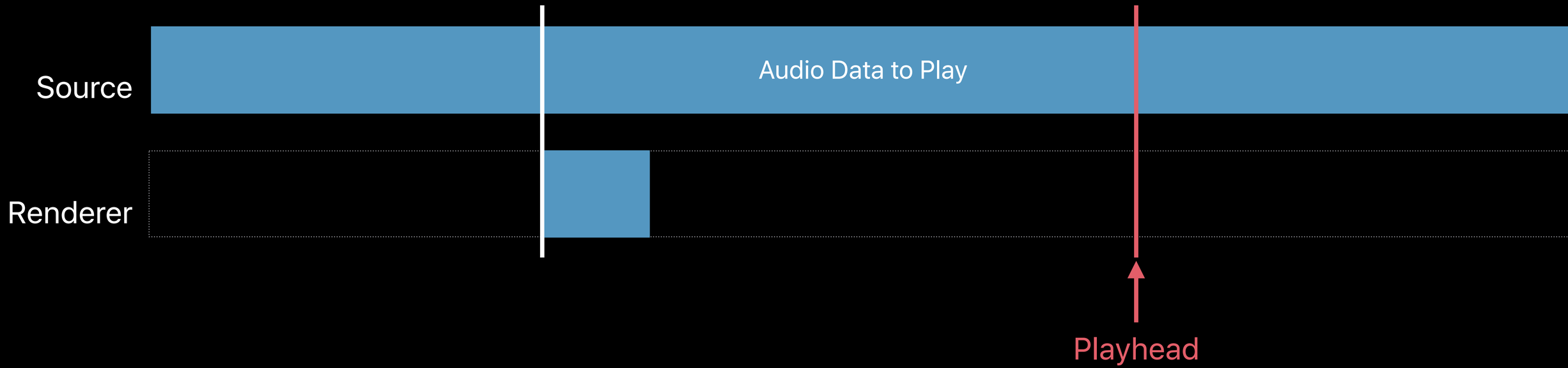
Seek

AVSampleBufferAudioRenderer



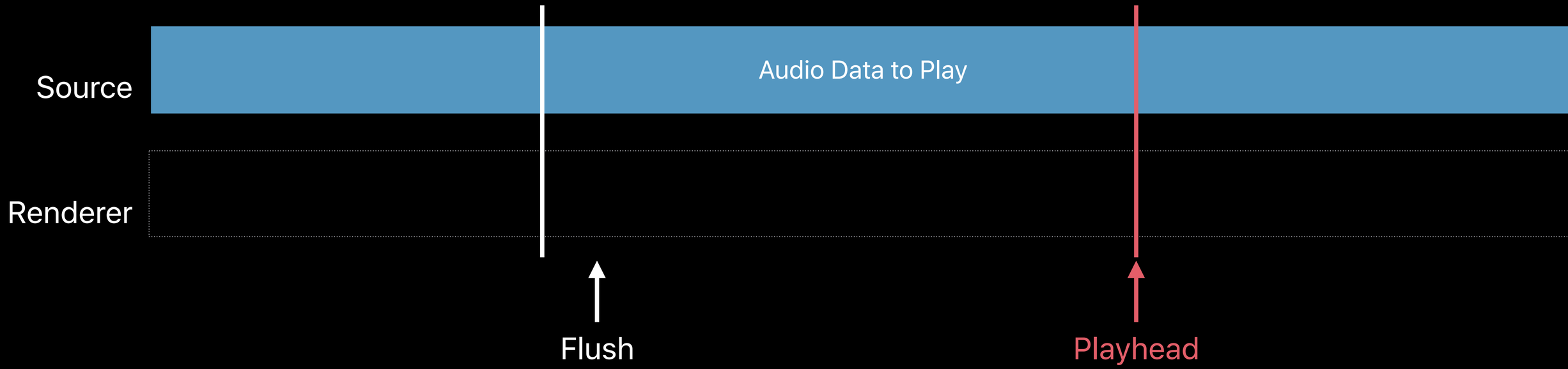
Seek

AVSampleBufferAudioRenderer



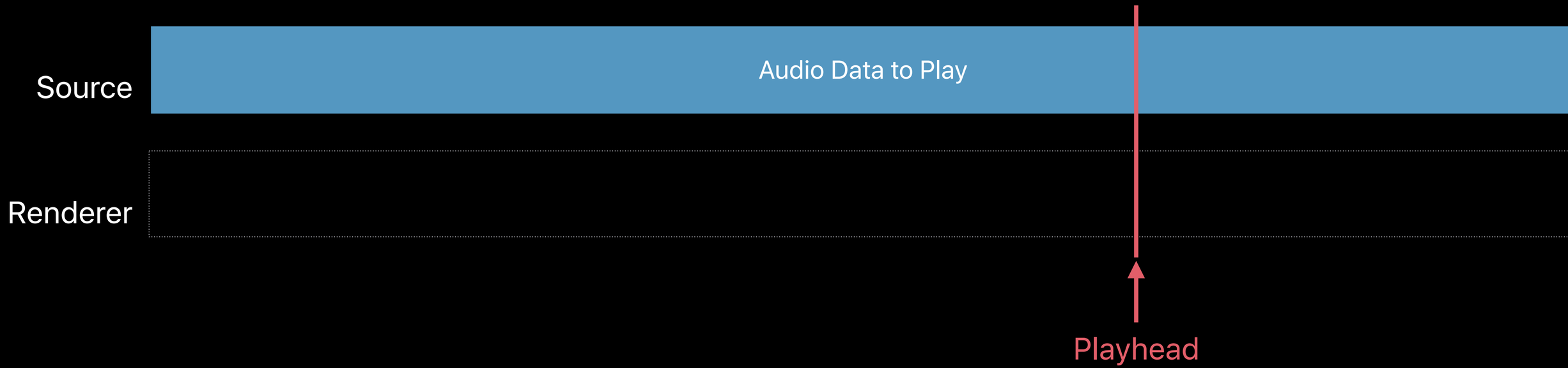
Seek

AVSampleBufferAudioRenderer



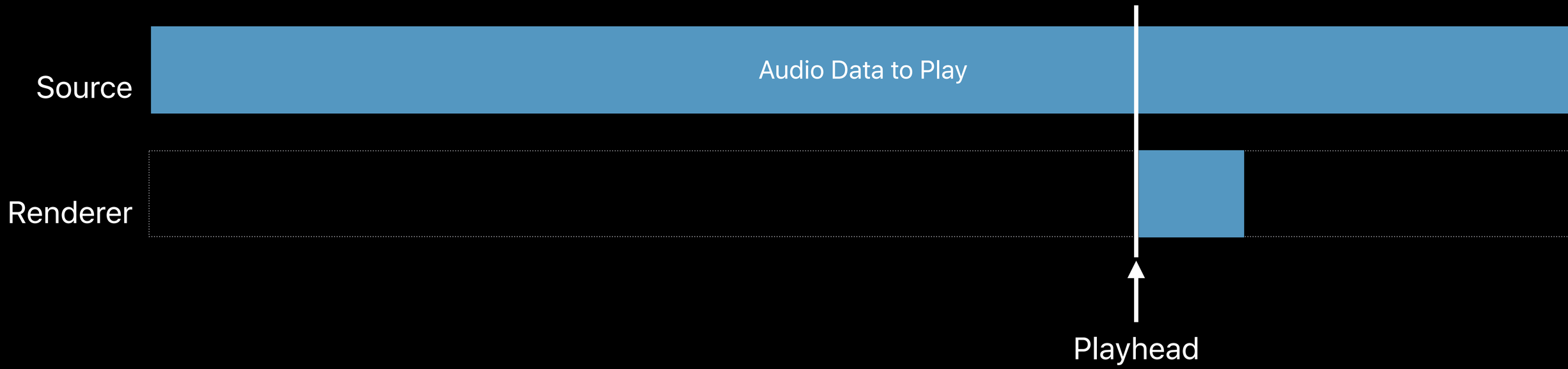
Seek

AVSampleBufferAudioRenderer



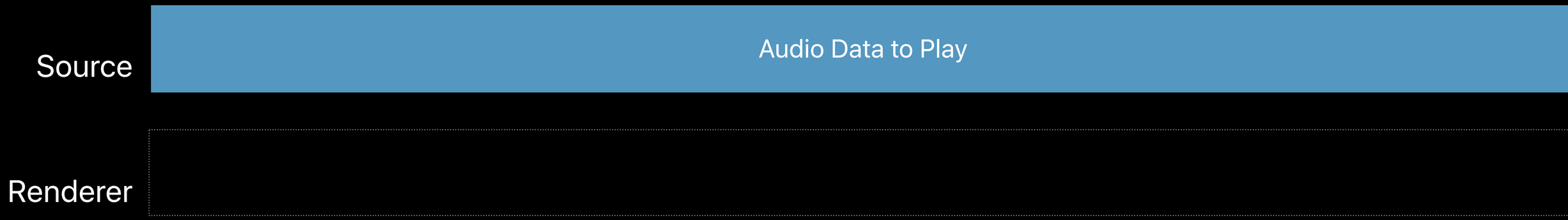
Seek

AVSampleBufferAudioRenderer



Seek

AVSampleBufferAudioRenderer



```
// 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)
}
```

```
// 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)
}
```

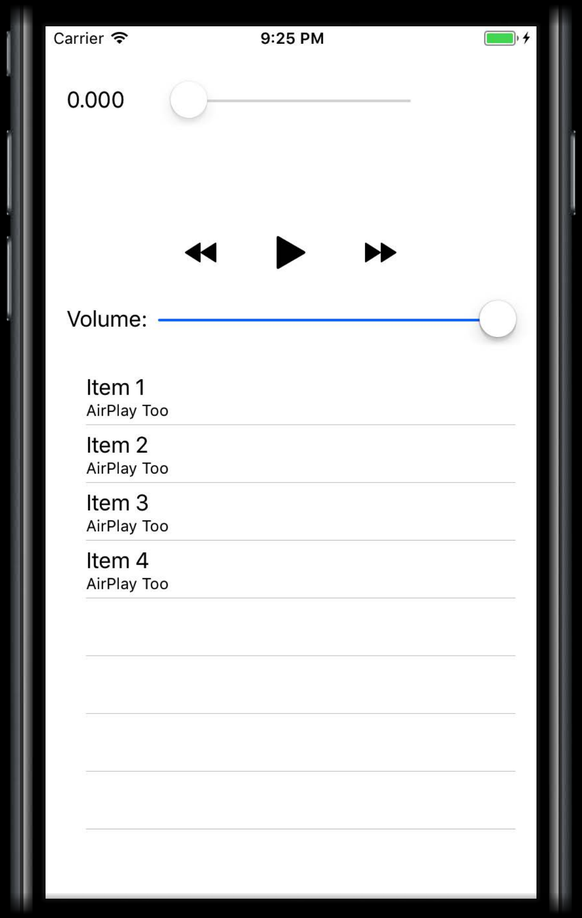
```
// Seek - AVSampleBufferAudioRenderer

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

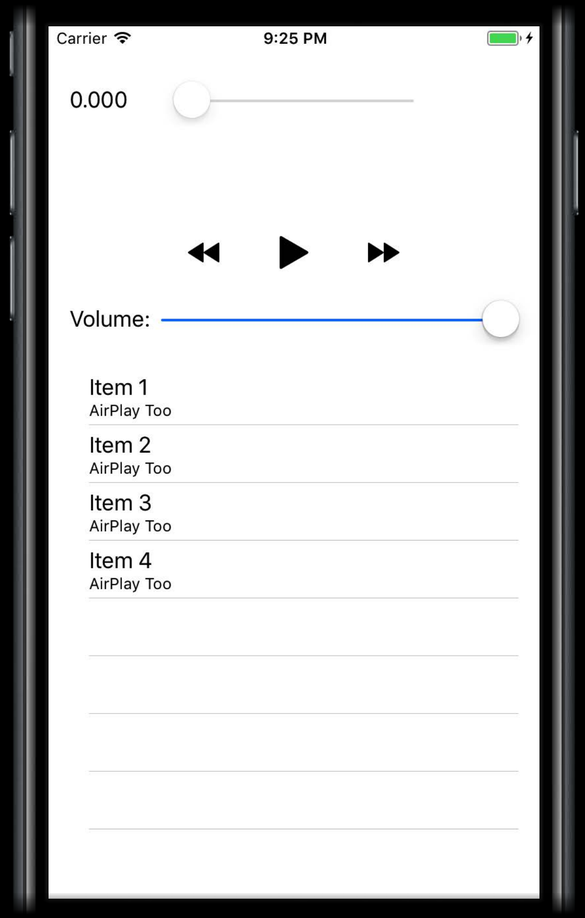
AVSampleBufferAudioRenderer

Play Queues

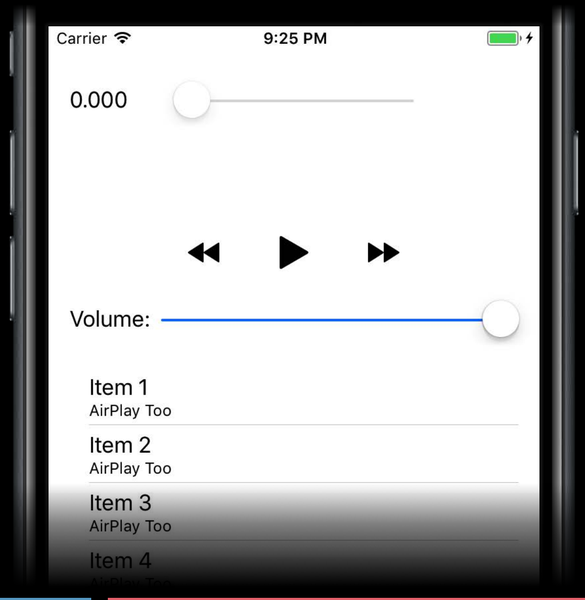

Carrier
9:25 PM
0.000
Volume:
Item 1
AirPlay Too
Item 2
AirPlay Too
Item 3
AirPlay Too
Item 4
AirPlay Too

Play Queues

Queue



Play Queues



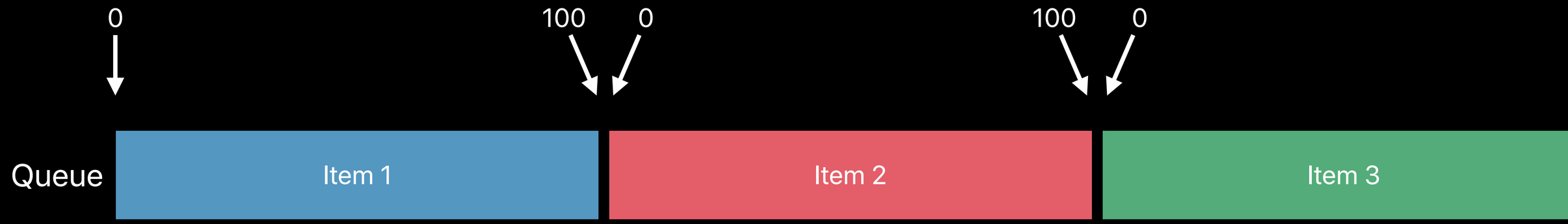
Queue | Item 1 | Item 2 | Item 3

Play Queues

Timelines

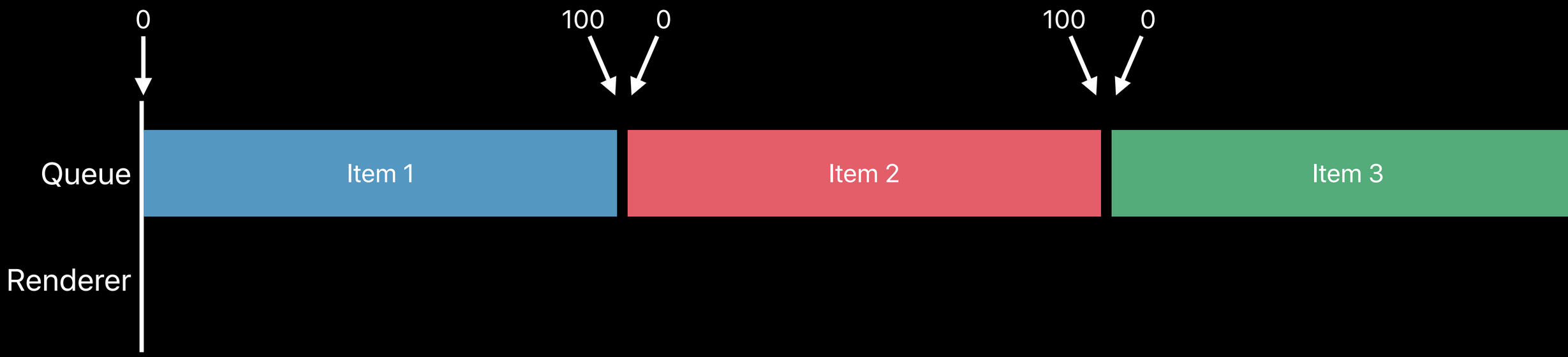
Play Queues

Timelines



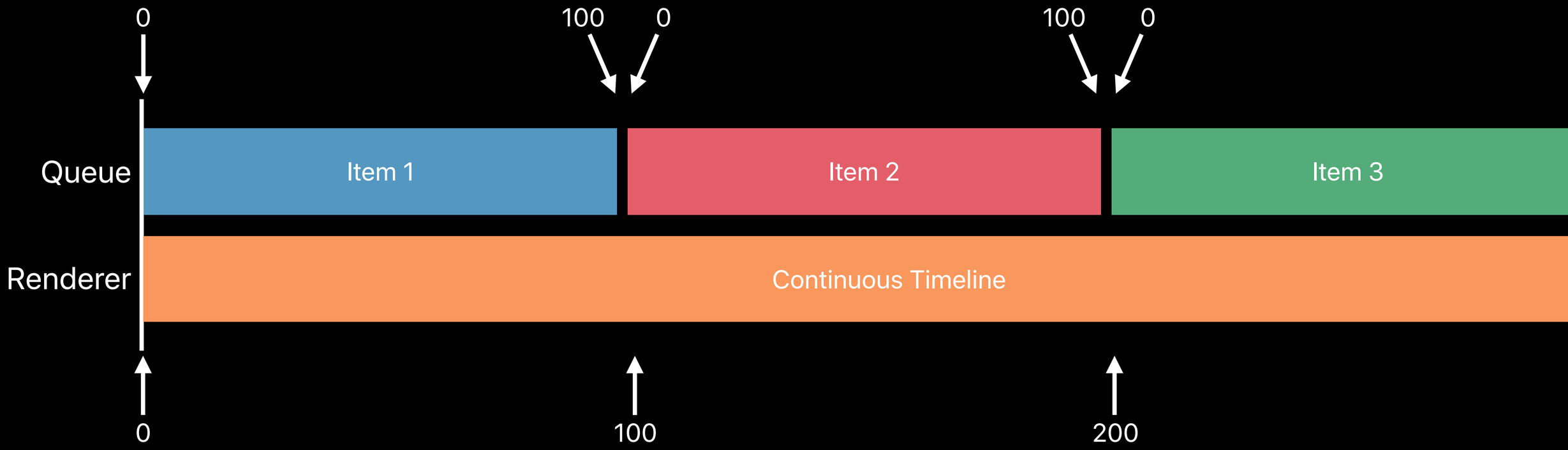
Play Queues

Timelines



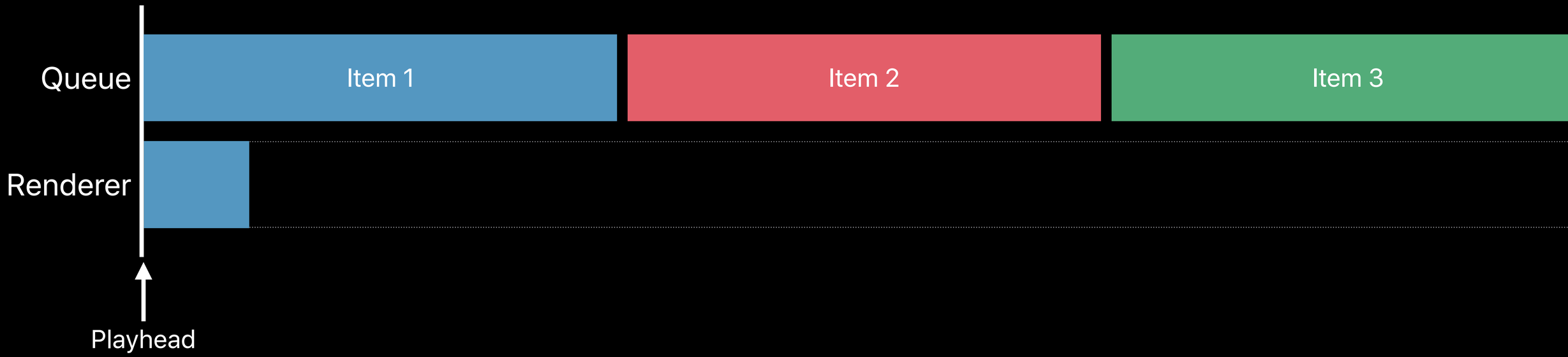
Play Queues

Timelines



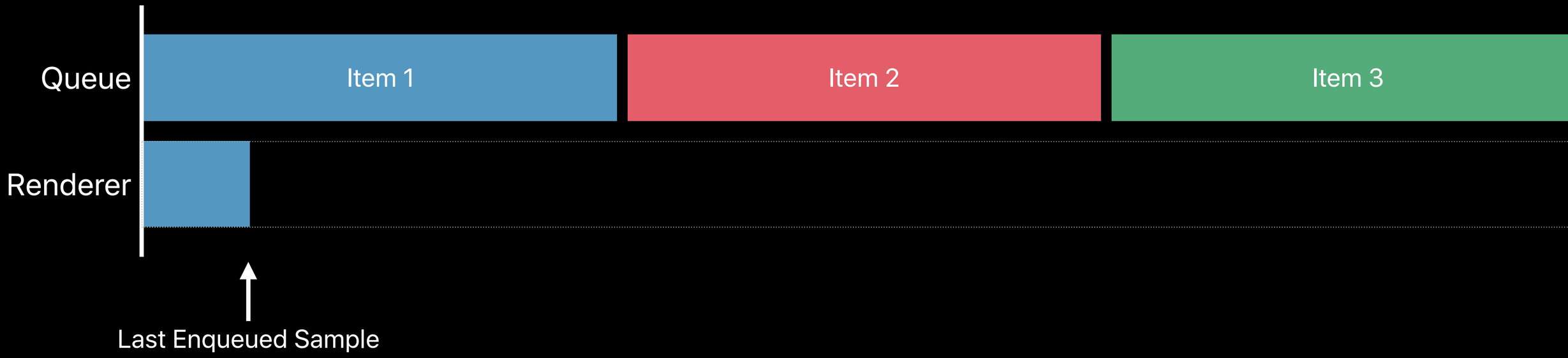
Play Queues

Enqueuing Audio Renderer



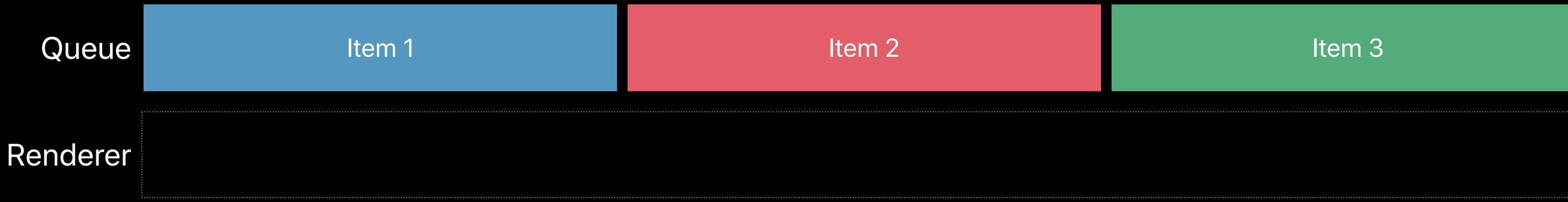
Play Queues

Enqueuing Audio Renderer



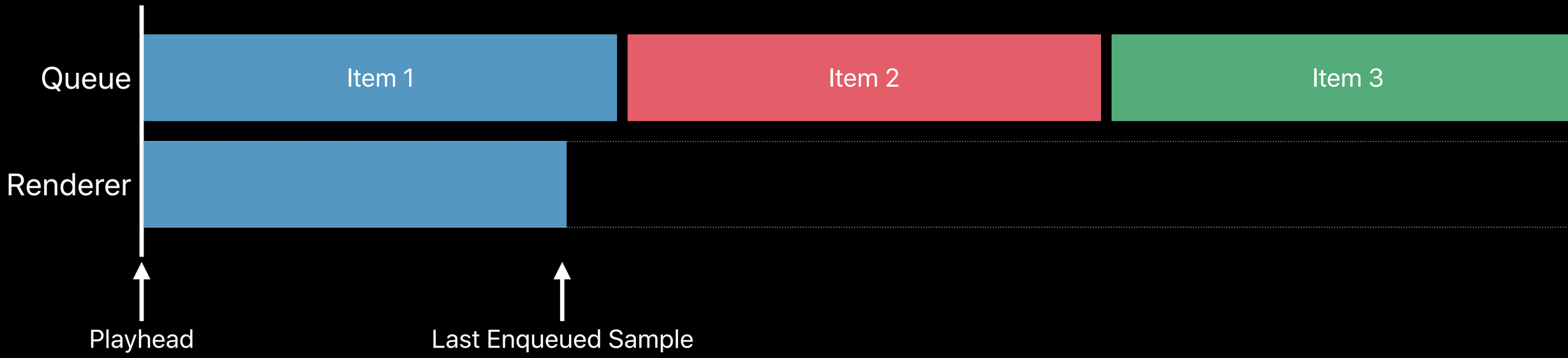
Play Queues

Enqueuing Audio Renderer



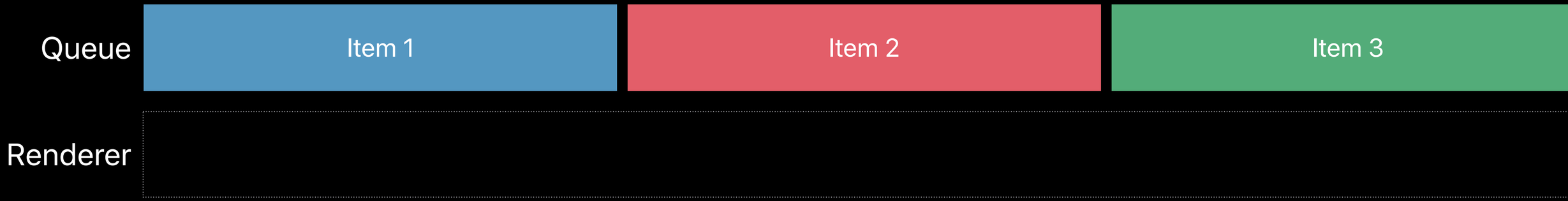
Play Queues

Deep Audio Buffer Levels



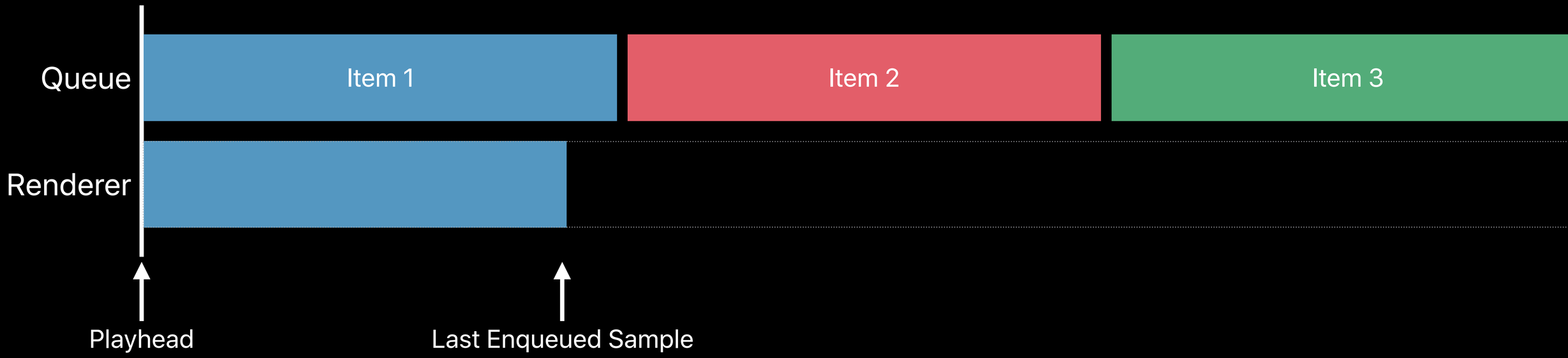
Play Queues

Deep Audio Buffer Levels



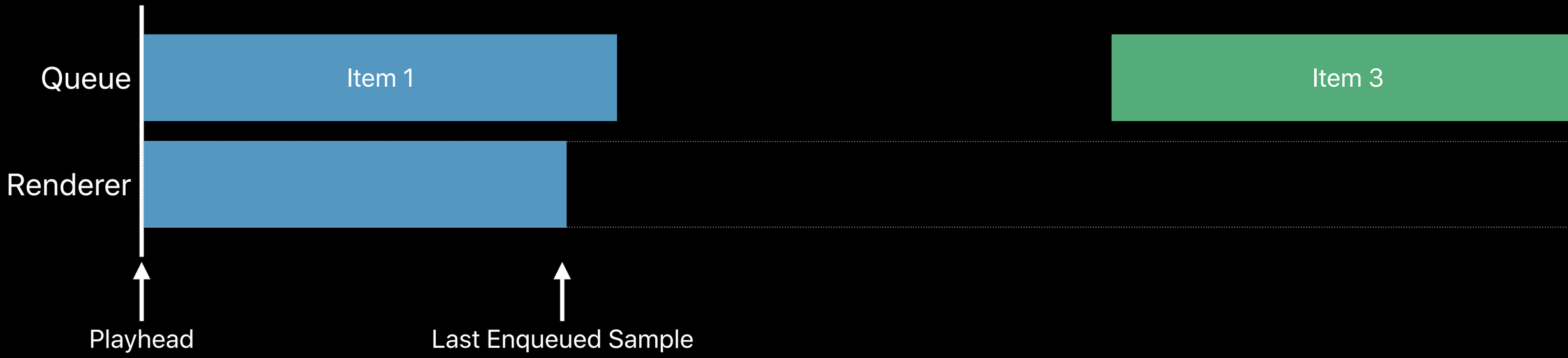
Play Queues

Editing



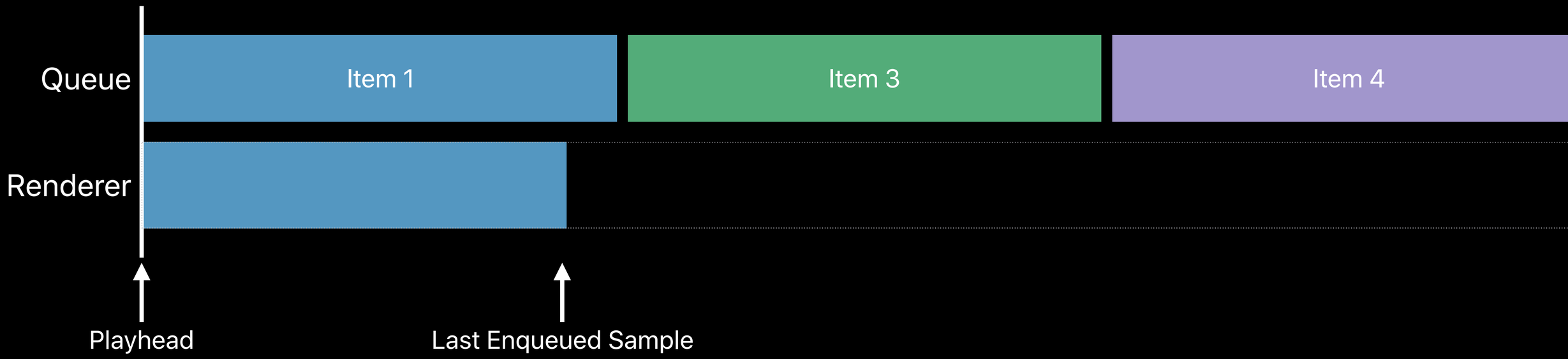
Play Queues

Editing



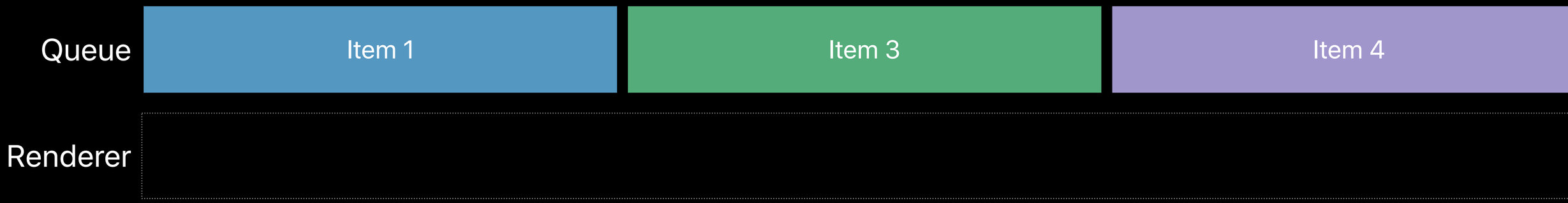
Play Queues

Editing



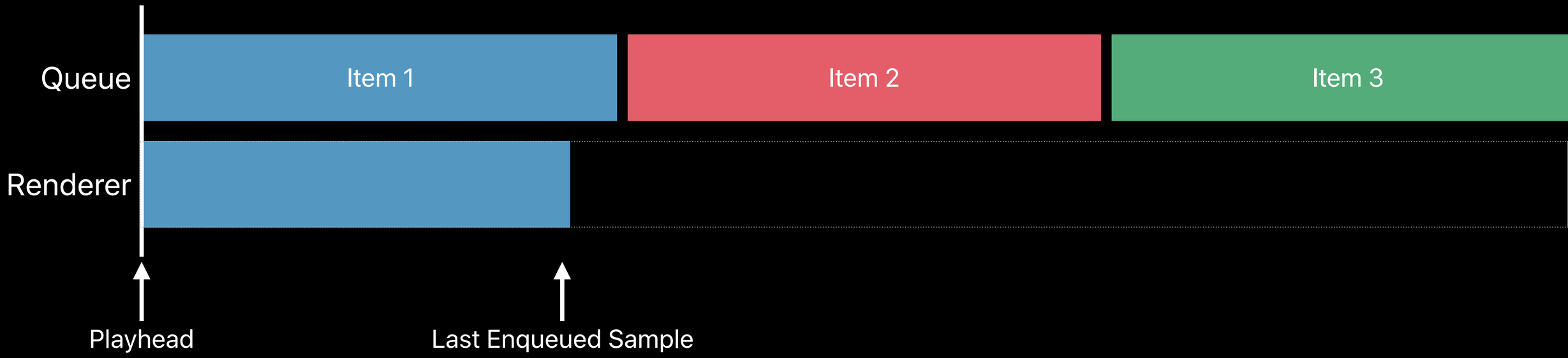
Play Queues

Editing



Play Queues

Editing during playback



Play Queues

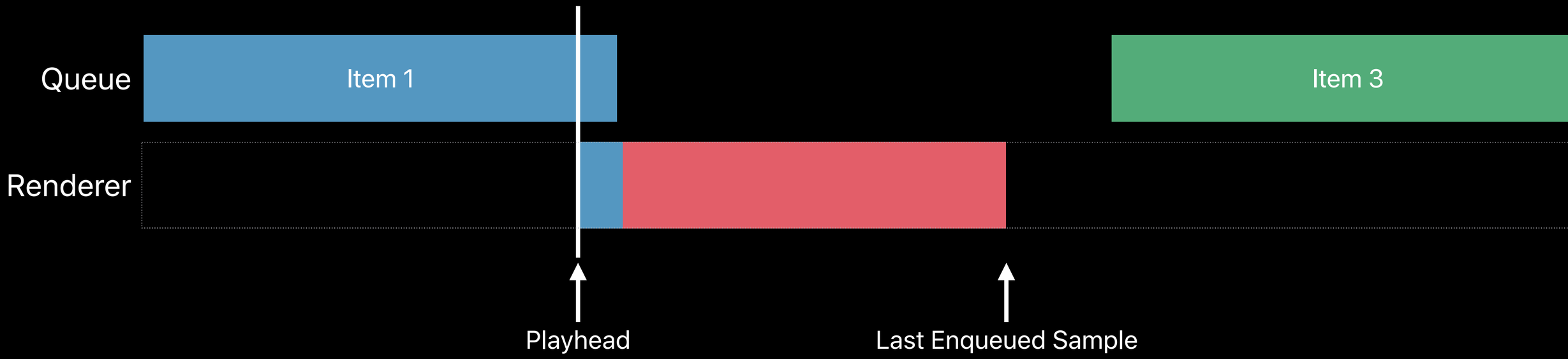
Editing during playback

Play Queues

Editing during playback

Play Queues

Editing during playback



Play Queues

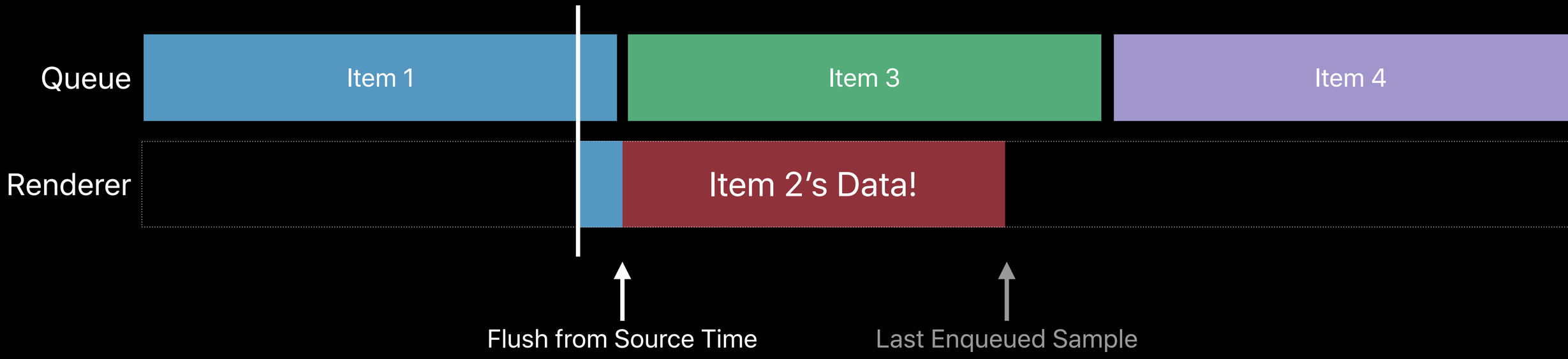
Editing during playback

Play Queues

Editing during playback

Play Queues

Replacing incorrect renderer data



Play Queues

Replacing incorrect renderer data

Play Queues

Replacing incorrect renderer data

Play Queues

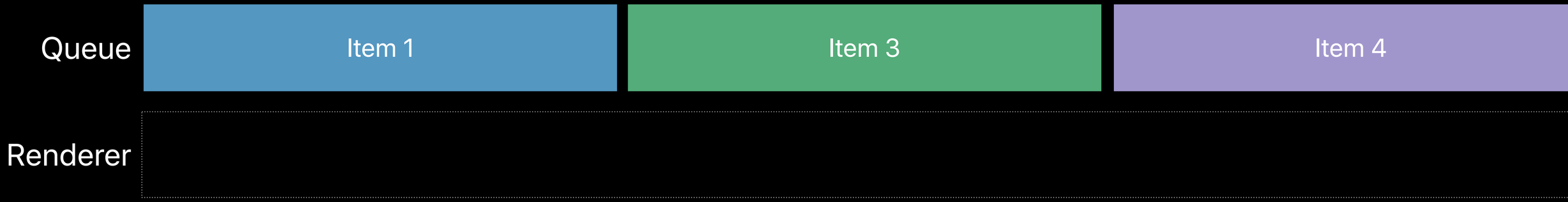
Replacing incorrect renderer data

Play Queues

Replacing incorrect renderer data

Play Queues

Replacing incorrect renderer data



Flushing from a Source Time

Flushing from a Source Time

1. Stop enqueueing audio data

Flushing from a Source Time

1. Stop enqueueing audio data
2. `flush(fromSourceTime:)`

Flushing from a Source Time

1. Stop enqueueing audio data
2. `flush(fromSourceTime:)`
3. Wait for the callback

Flushing from a Source Time

Gotchas

Flushing from a Source Time

Gotchas

Flush may fail!

Flushing from a Source Time

Gotchas

Flush may fail!

- Source time is too close to playhead

Flushing from a Source Time

Gotchas

Flush may fail!

- Source time is too close to playhead

Wait for the callback!

```
// 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
        }
    }
}
```

```
// 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
        }
    }
}
```

```
// 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
        }
    }
}
```

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

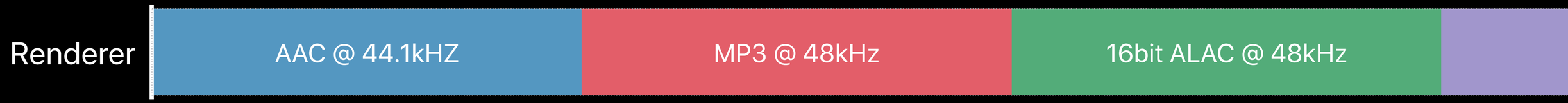
Supported Audio Formats

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



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)

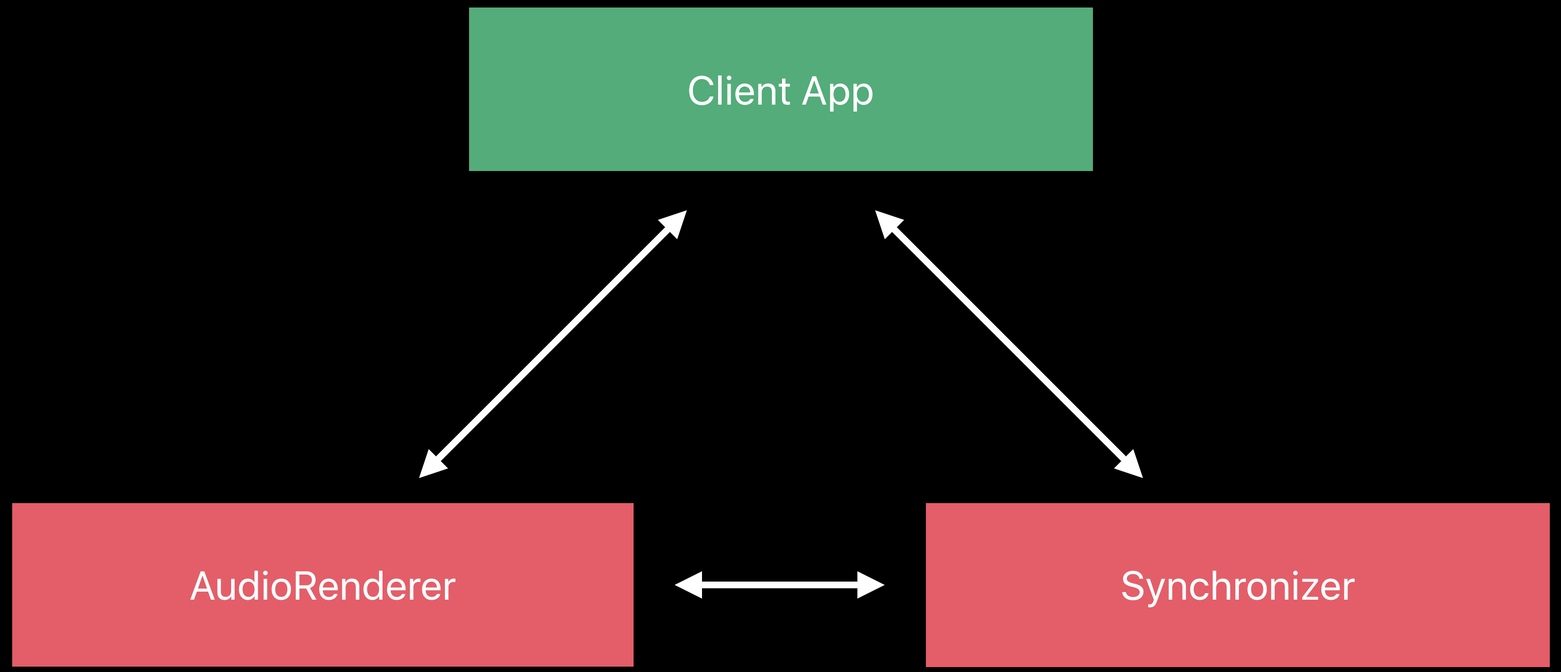
Interleaved channel formats

1-2 seconds of audio per `CMSampleBuffer`

Video Synchronization

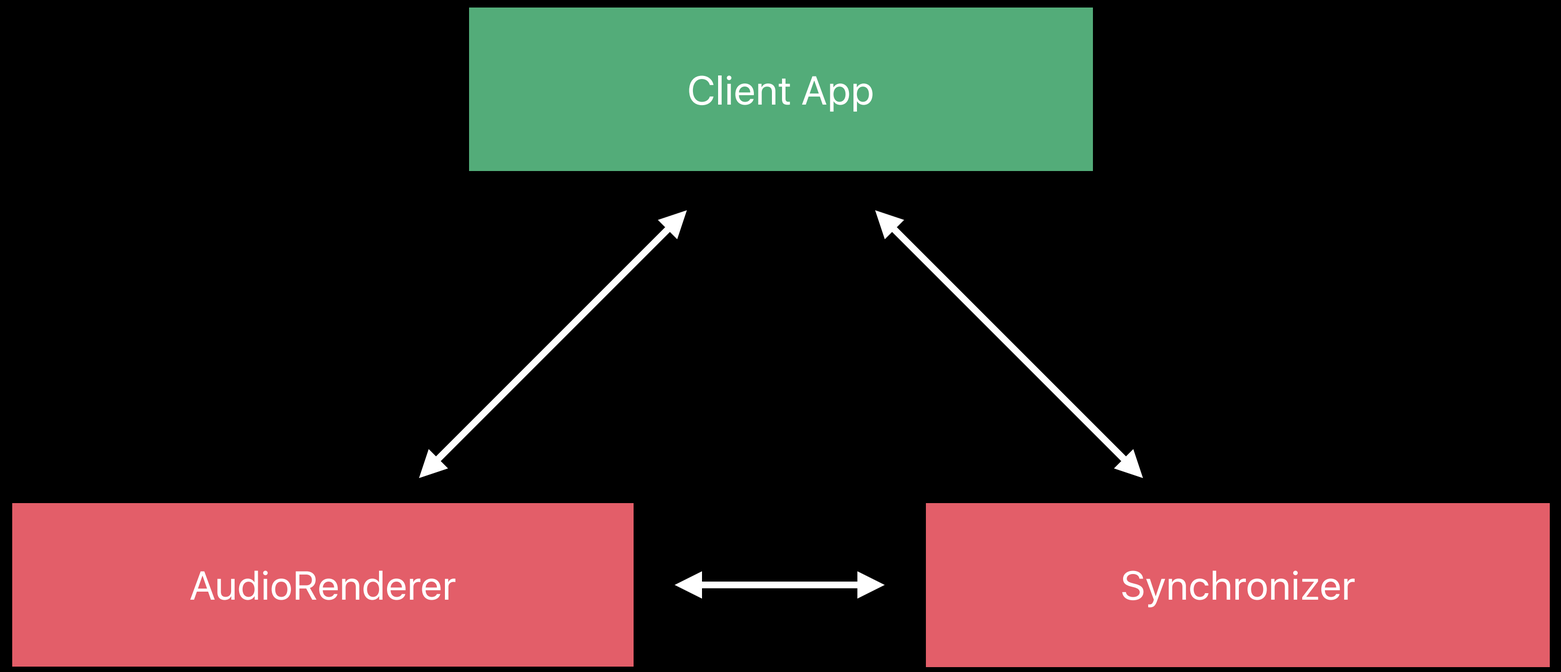
AVSampleBufferDisplayLayer

Video Synchronization



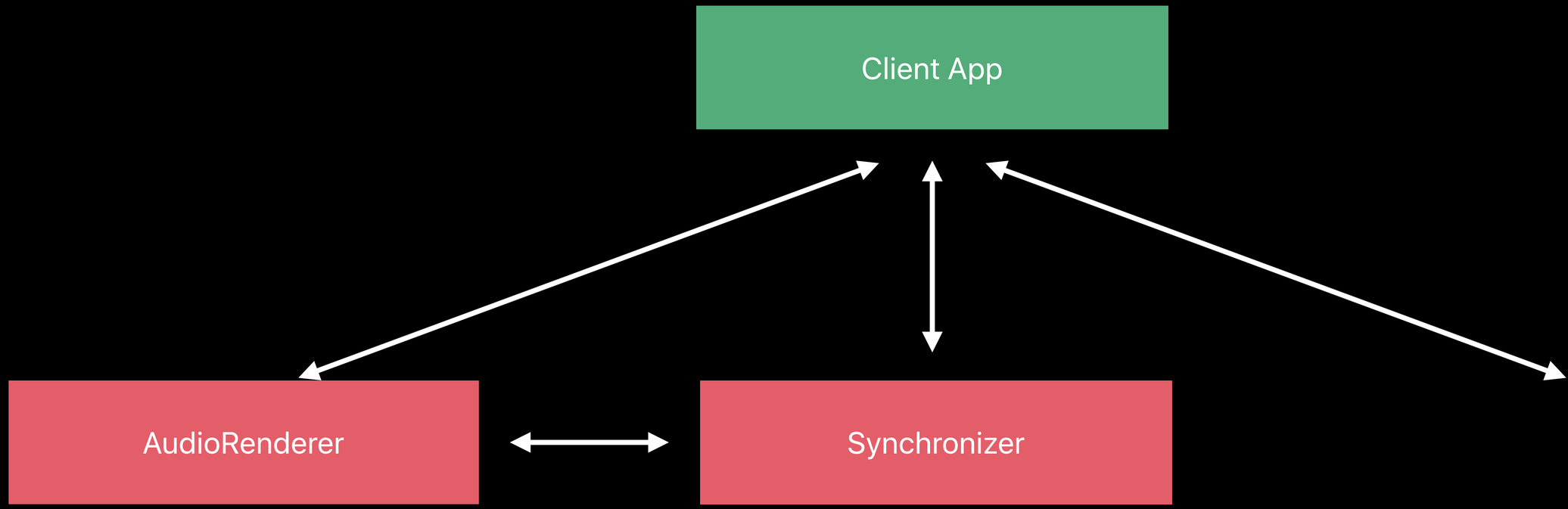
Video Synchronization

AVSampleBufferDisplayLayer



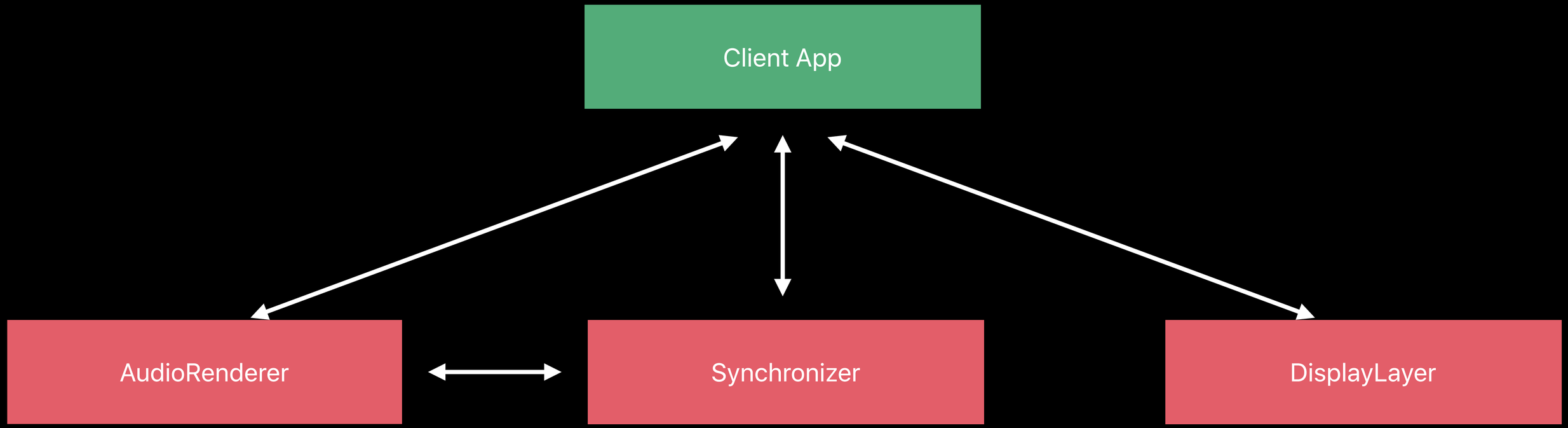
Video Synchronization

AVSampleBufferDisplayLayer



Video Synchronization

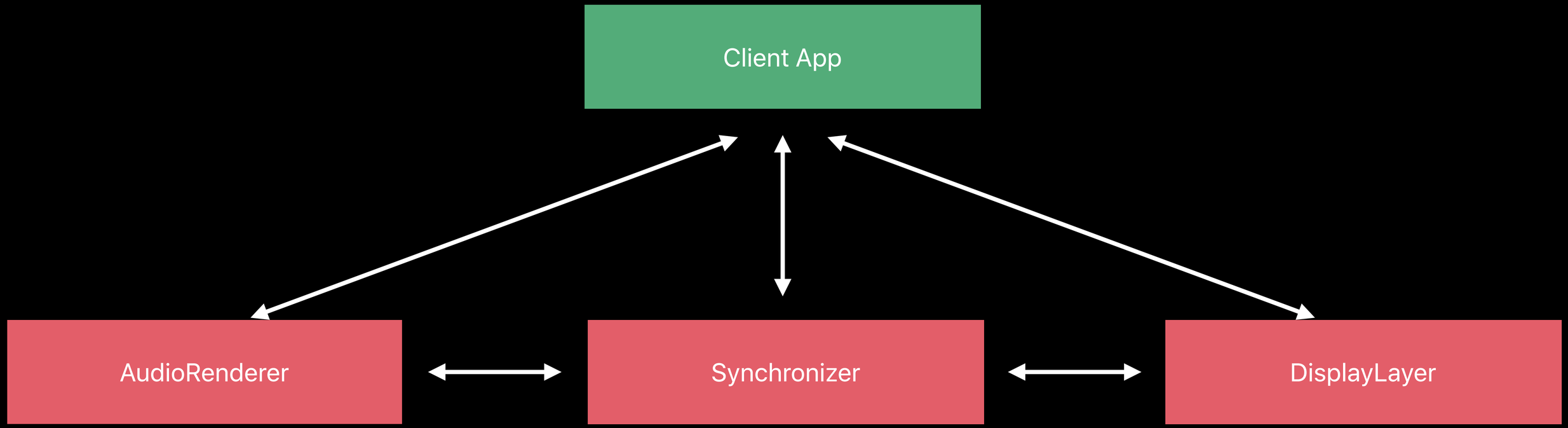
AVSampleBufferDisplayLayer



Video Synchronization

AVSampleBufferDisplayLayer

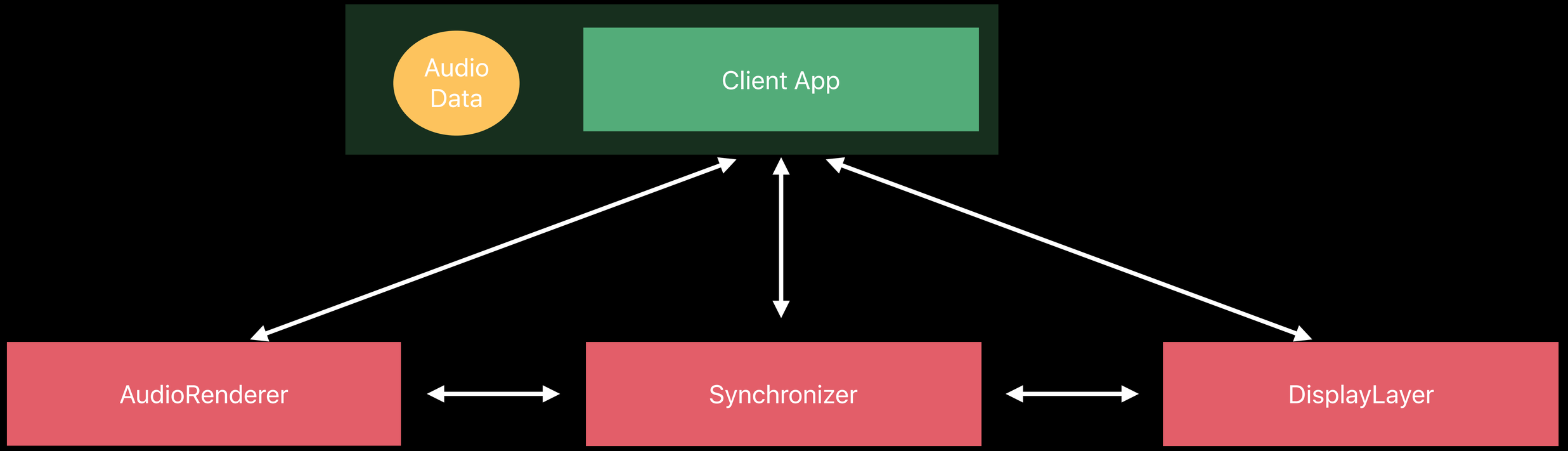
NEW



Video Synchronization

AVSampleBufferDisplayLayer

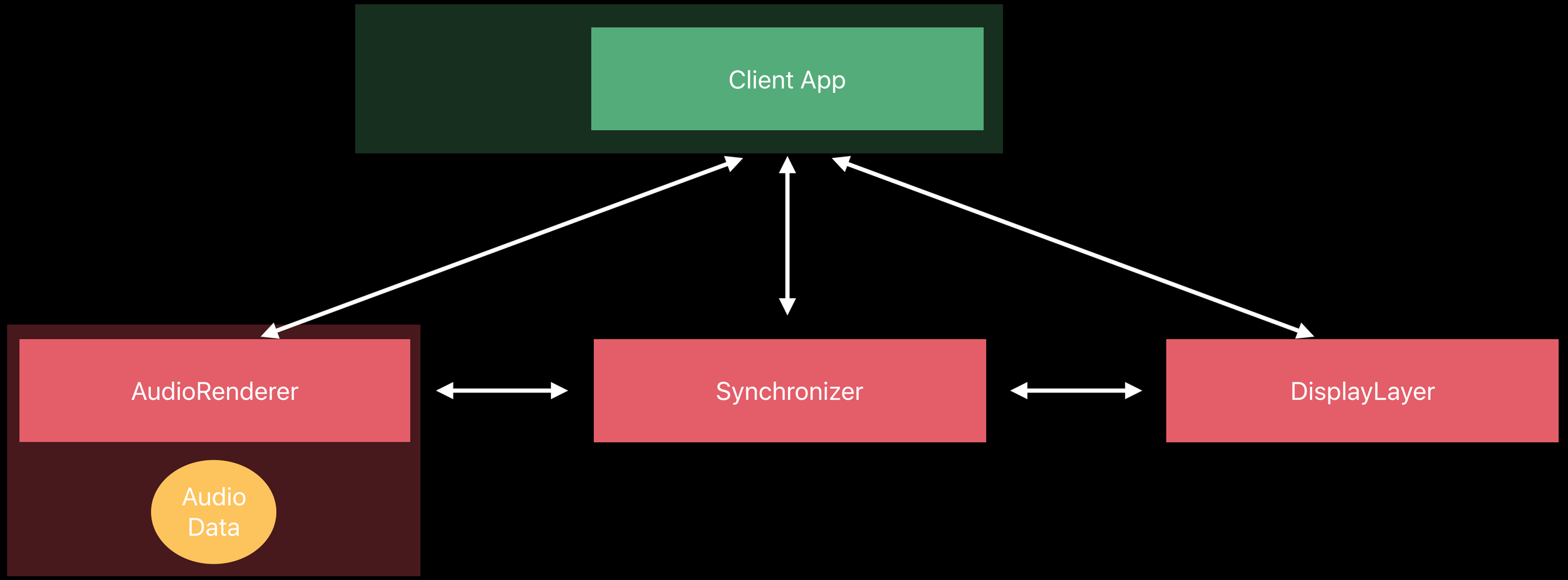
NEW



Video Synchronization

AVSampleBufferDisplayLayer

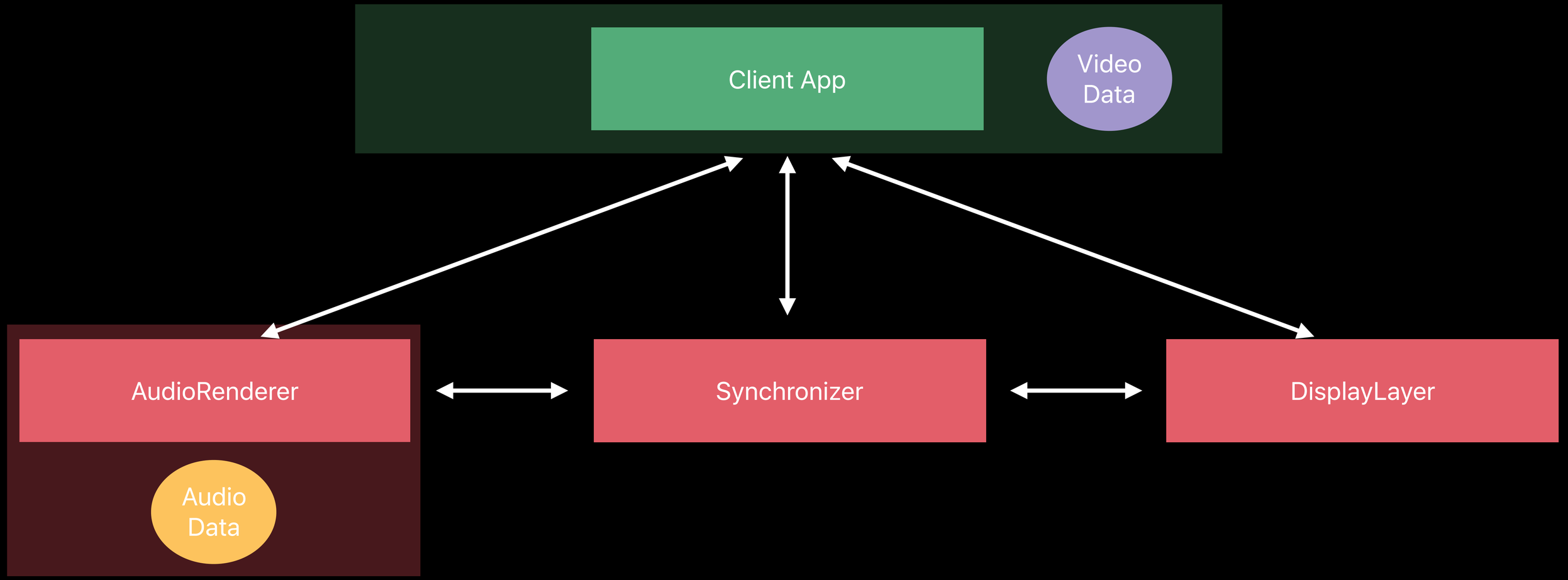
NEW



Video Synchronization

AVSampleBufferDisplayLayer

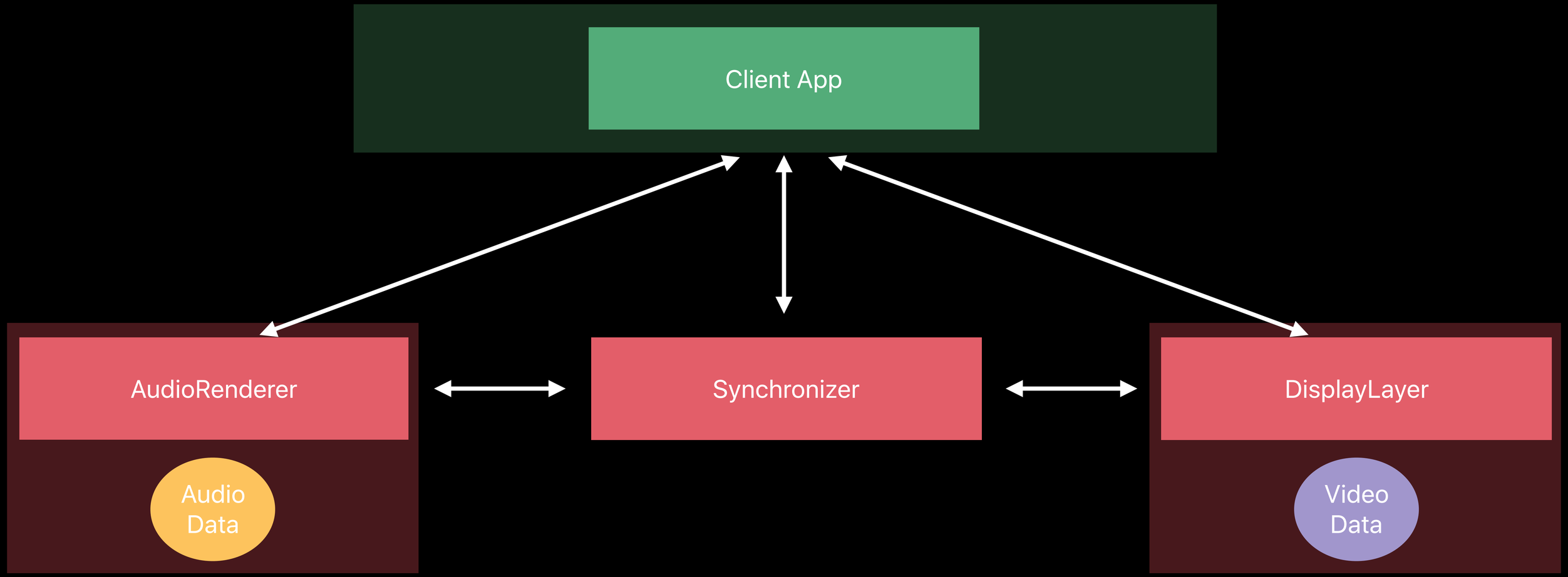
NEW



Video Synchronization

AVSampleBufferDisplayLayer

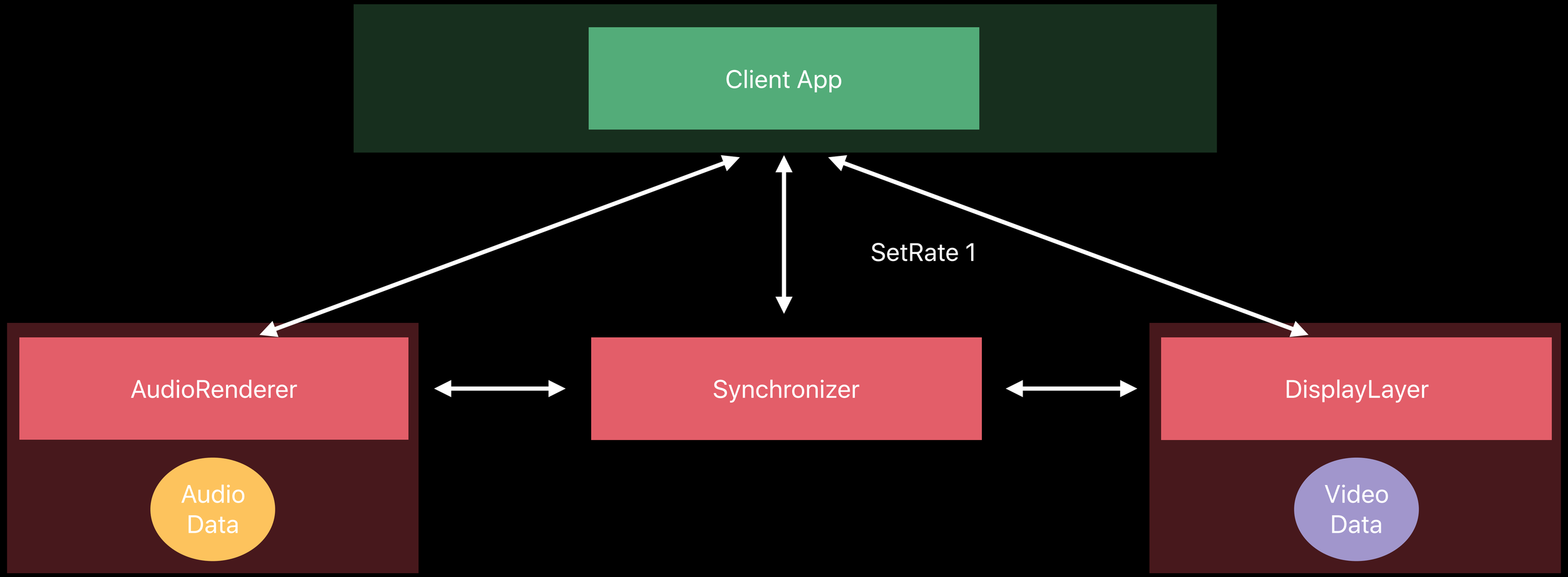
NEW



Video Synchronization

AVSampleBufferDisplayLayer

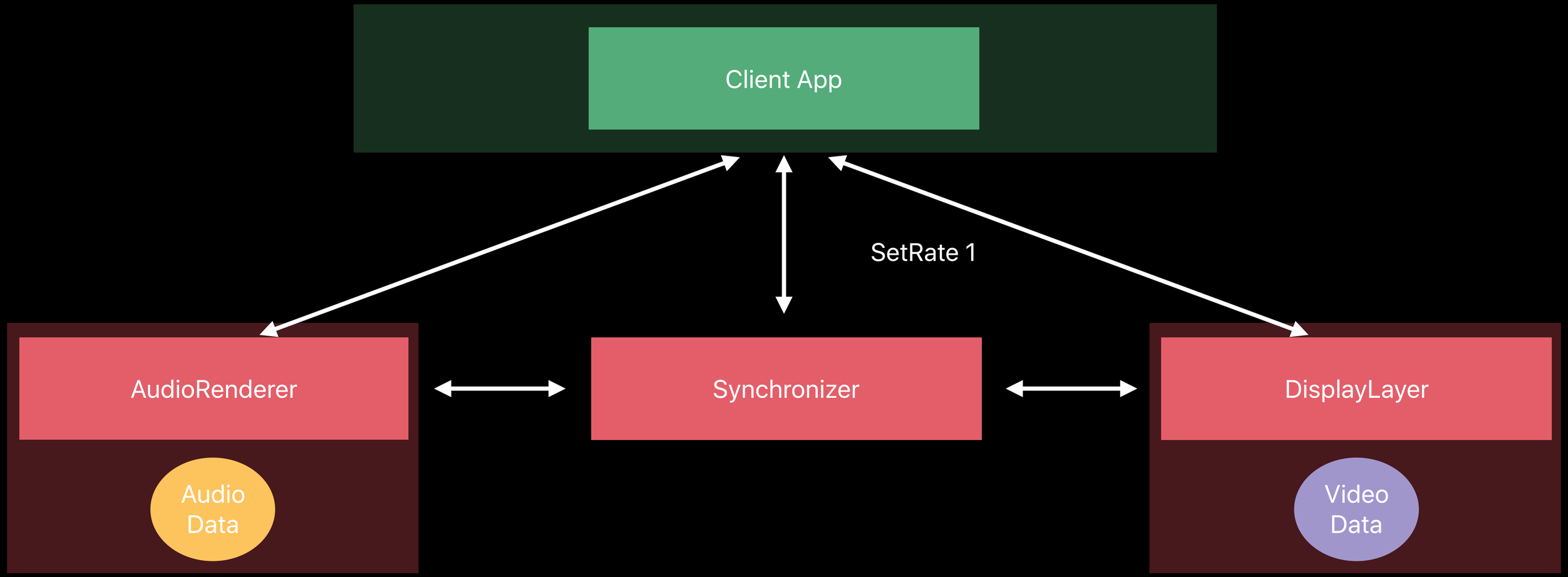
NEW



Video Synchronization

AVSampleBufferDisplayLayer

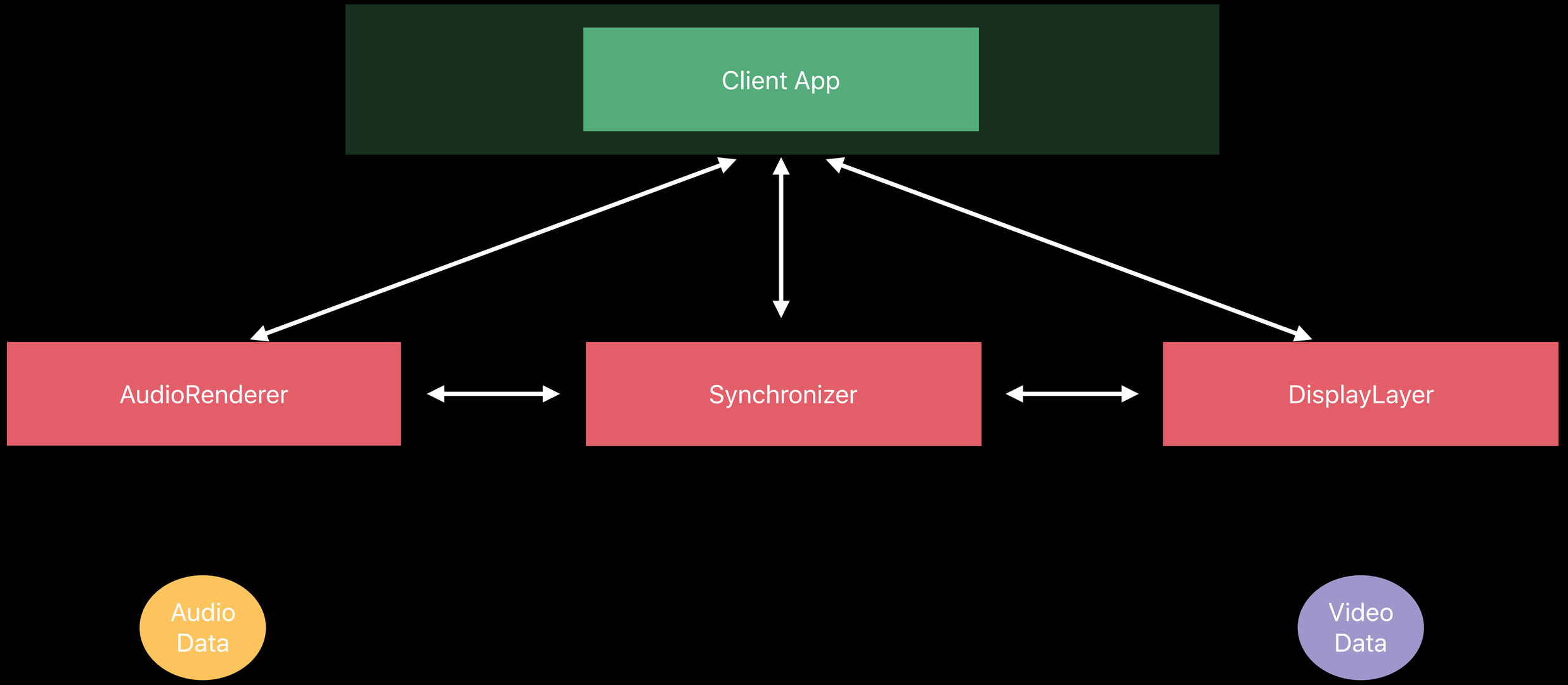
NEW



Video Synchronization

AVSampleBufferDisplayLayer

NEW



Availability

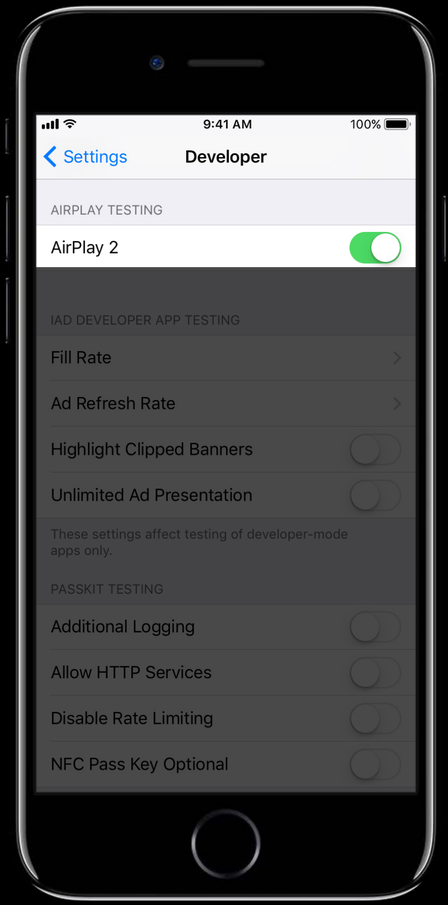
Availability of AirPlay 2

Availability of AirPlay 2

APIs and enhanced buffering	Beta 1

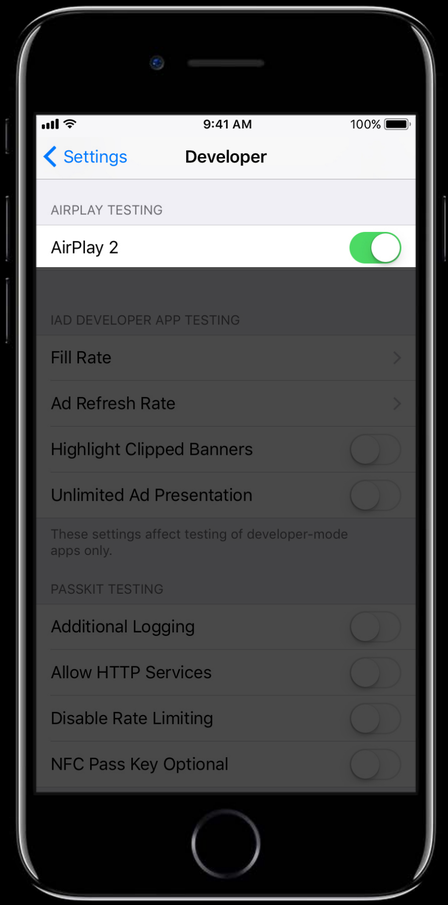
Availability of AirPlay 2

APIs and enhanced buffering	Beta 1



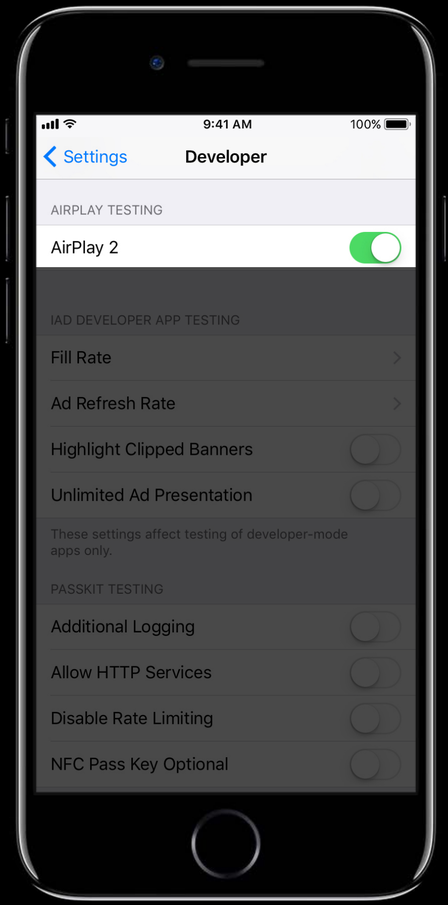
Availability of AirPlay 2

APIs and enhanced buffering	Beta 1
Multi-room audio	Upcoming Beta



Availability of AirPlay 2

APIs and enhanced buffering	Beta 1
Multi-room audio	Upcoming Beta
Available to users	Customer Release



Summary

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

https://developer.apple.com/wwdc17/509

Related Sessions

What’s New in Audio	WWDC 2017
Introducing MusicKit	WWDC 2017

Labs

AirPlay Lab	Technology Lab A	Fri 9:00AM-11:00AM


WWDC17