Advances in iOS Photography

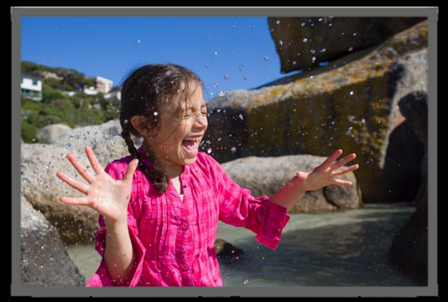
Live Photo, RAW, and Wide Color Capture with AV Foundation

Session 501

Brad Ford Apple





IPBERRY
HEE TEA
VANILLA













Past Sessions

developer.apple.com

What's New In Camera Capture (iOS 6)	WWDC 2012
What's New In Camera Capture (iOS 7)	WWDC 2013
Camera Capture Manual Controls (iOS 8 / Yosemite)	WWDC 2014

Agenda

Agenda

New AVCaptureOutput

Agenda

New AVCaptureOutput

New photography-driven features in iOS 10!

Agenda

New AVCaptureOutput

New photography-driven features in iOS 10!

- Live Photos

Agenda

New AVCaptureOutput

New photography-driven features in iOS 10!

- Live Photos
- RAW and DNG

Agenda

New AVCaptureOutput

New photography-driven features in iOS 10!

- Live Photos
- RAW and DNG
- Preview (Thumbnail) images

Agenda

New AVCaptureOutput

New photography-driven features in iOS 10!

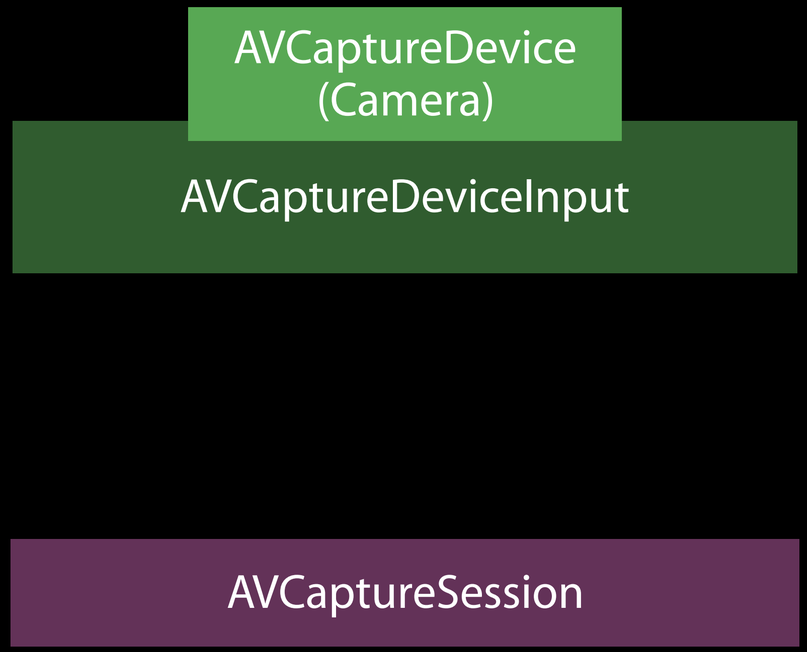
- Live Photos
- RAW and DNG
- Preview (Thumbnail) images
- Wide Color

Review: AV Foundation Capture Objects

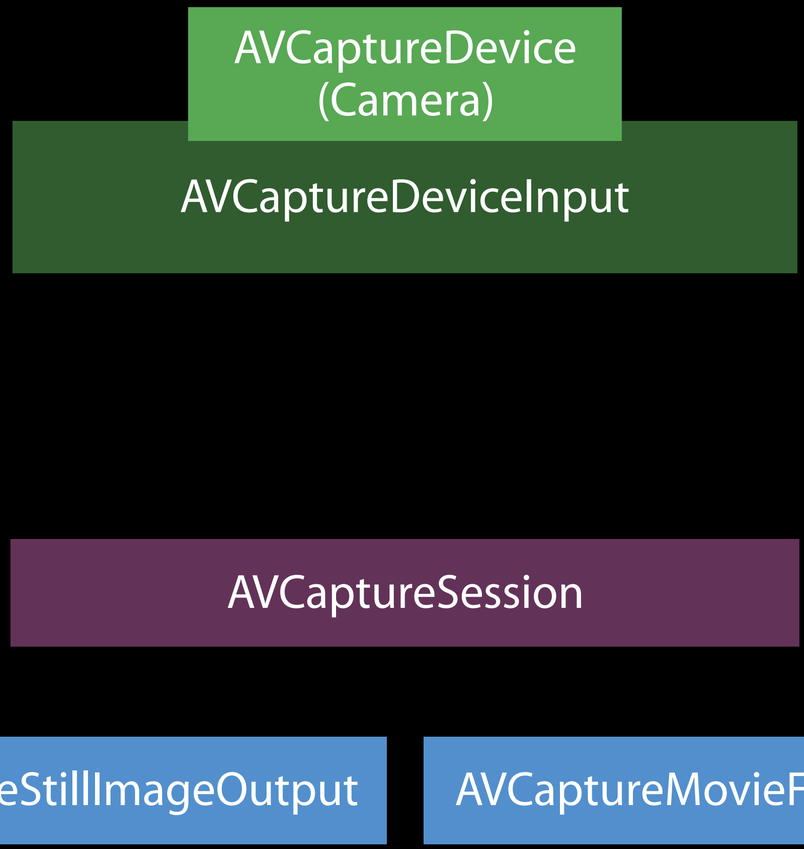
Review: AV Foundation Capture Objects

AVCaptureSession

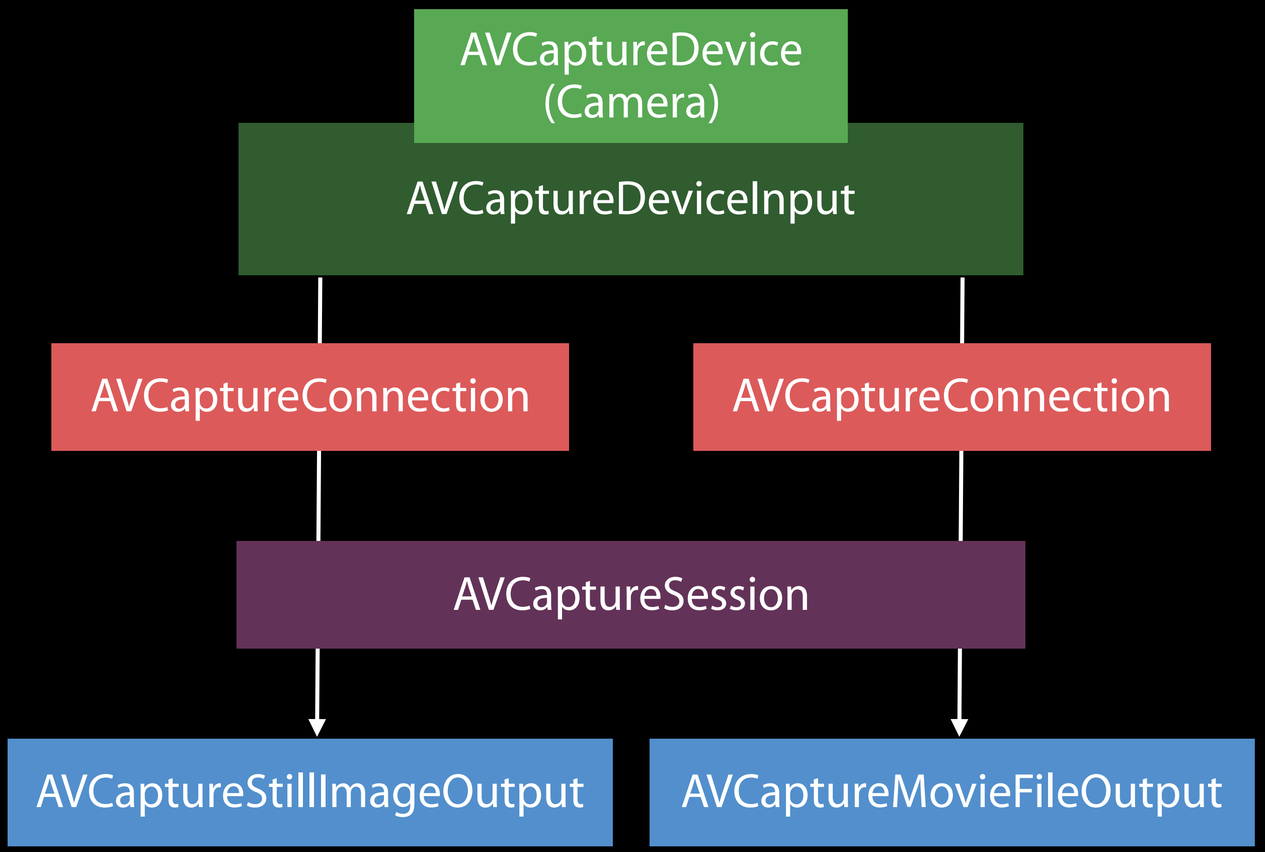
Review: AV Foundation Capture Objects



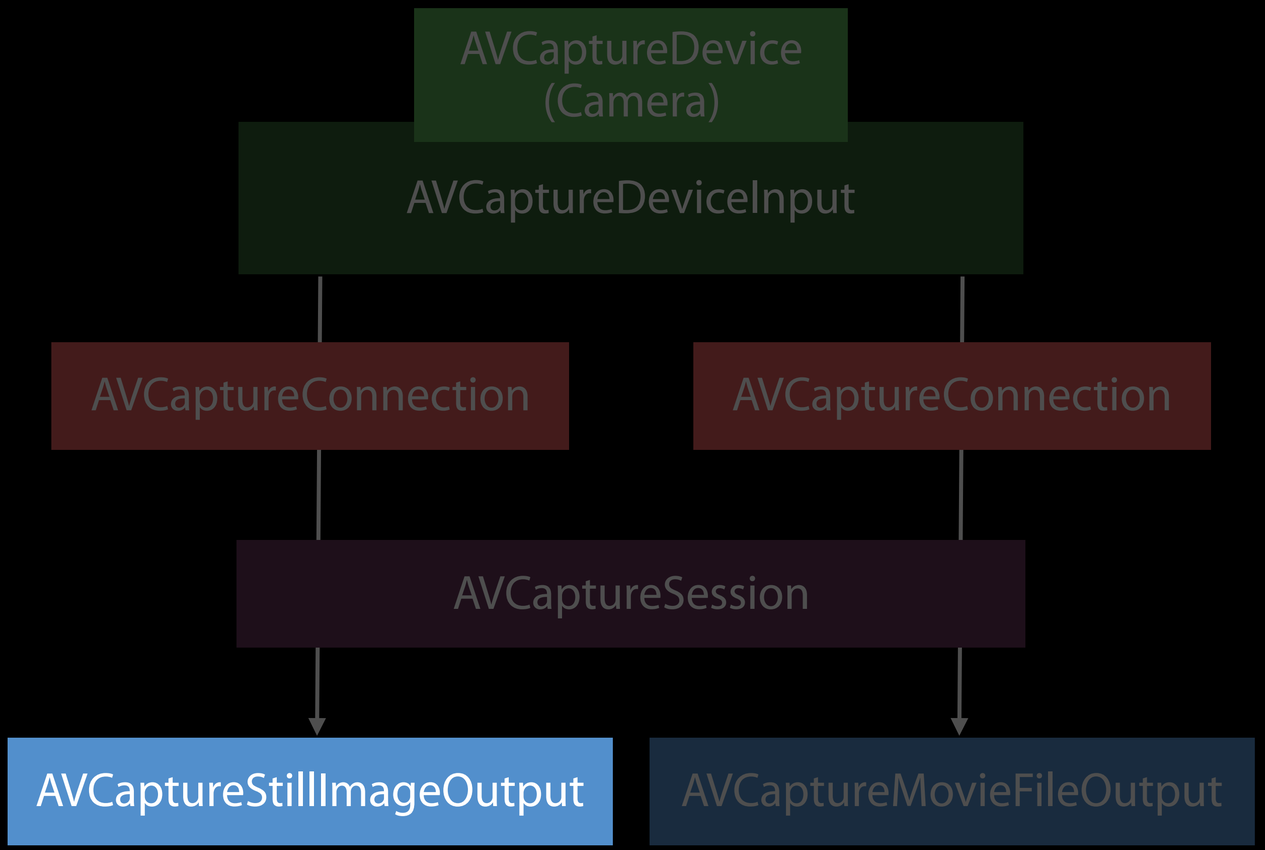
Review: AV Foundation Capture Objects



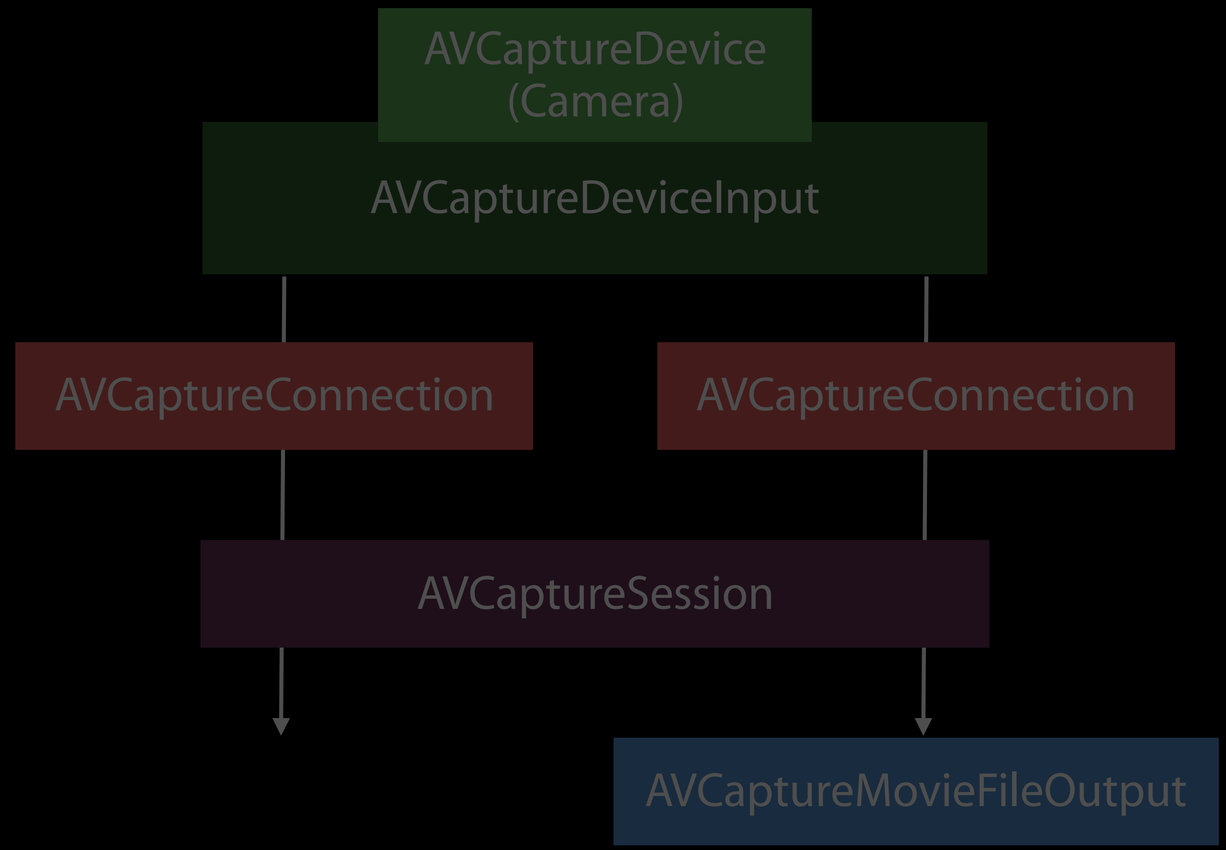
Review: AV Foundation Capture Objects



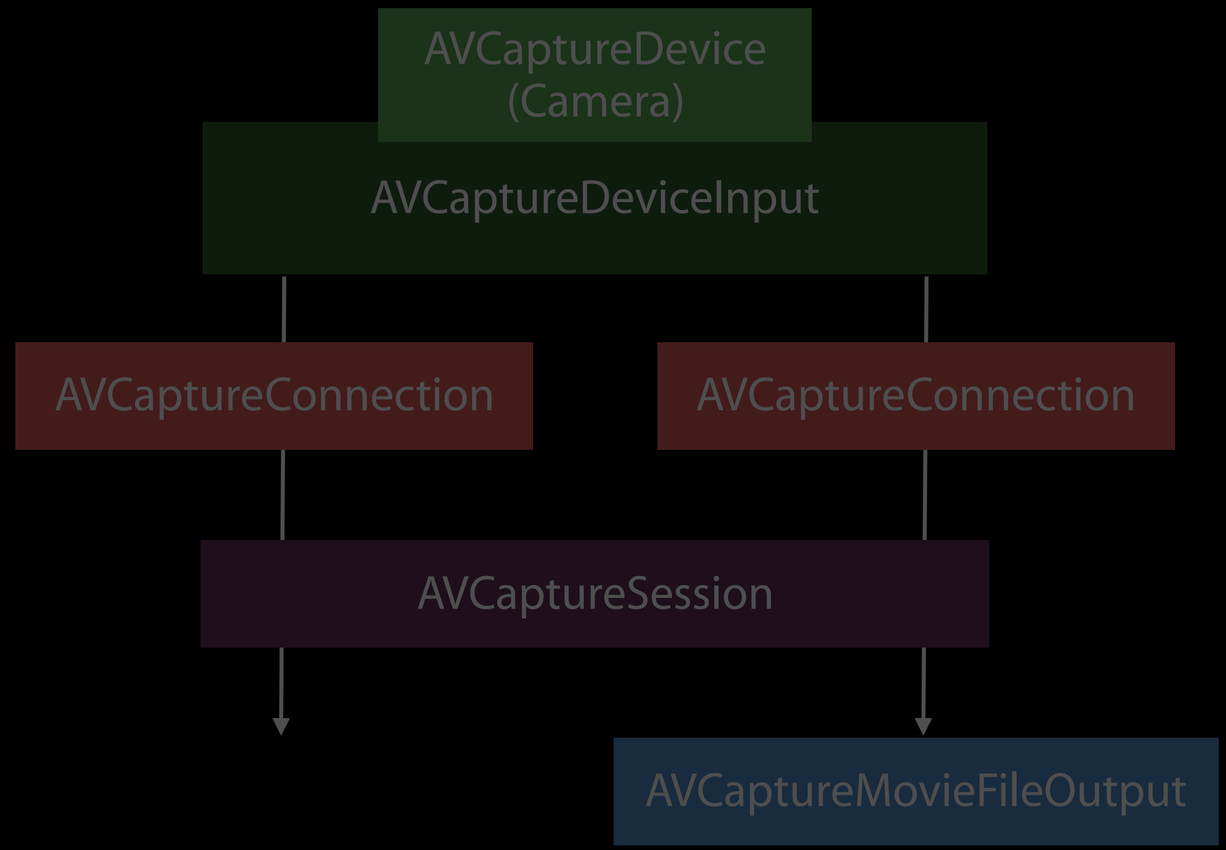
Review: AV Foundation Capture Objects



Review: AV Foundation Capture Objects

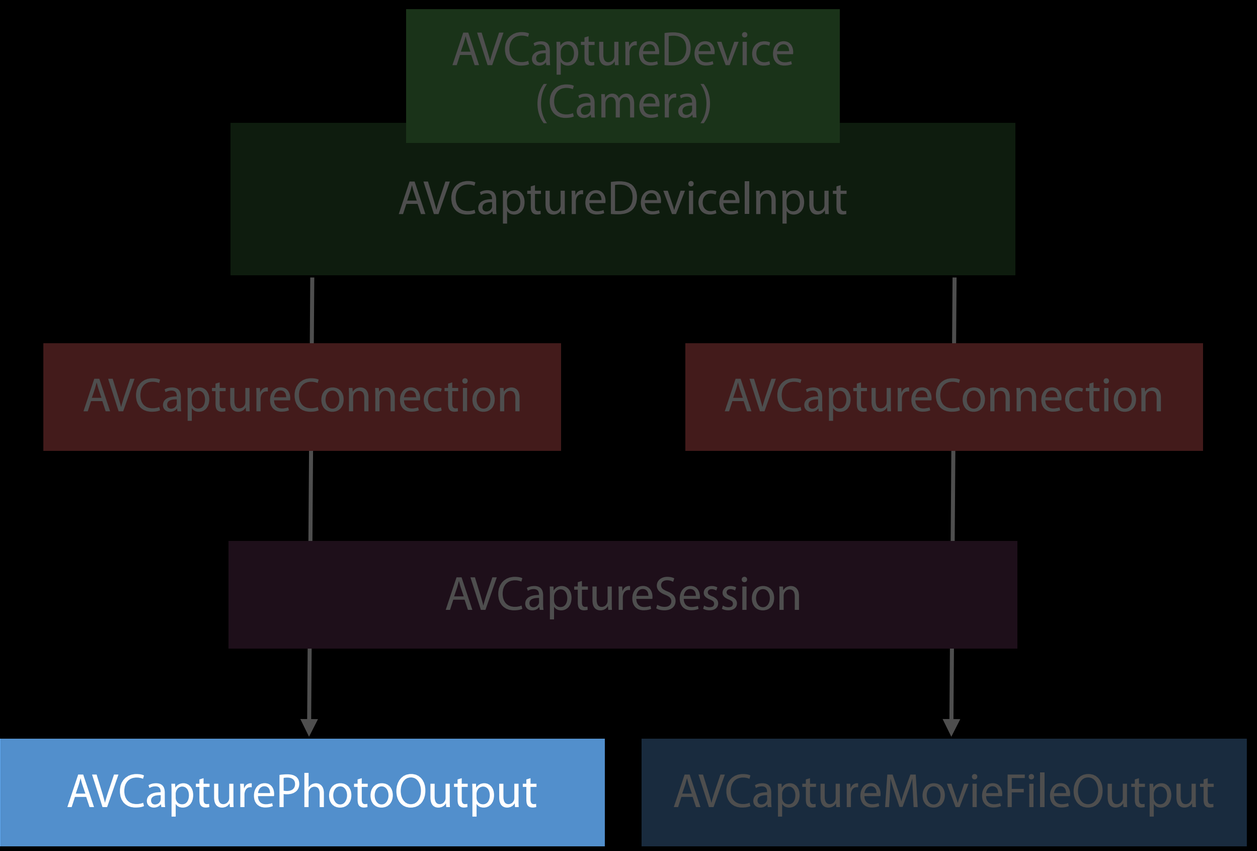


Review: AV Foundation Capture Objects



Review: AV Foundation Capture Objects

NEW



AVCapturePhotoOutput Design Features

NEW

AVCapturePhotoOutput Design Features

NEW

Functional programming model

AVCapturePhotoOutput Design Features

NEW

Functional programming model

Photo settings encapsulation

AVCapturePhotoOutput Design Features

NEW

Functional programming model

Photo settings encapsulation

A delegate-style interface for tracking the progress of photo capture requests

AVCapturePhotoOutput Design Features

NEW

Functional programming model

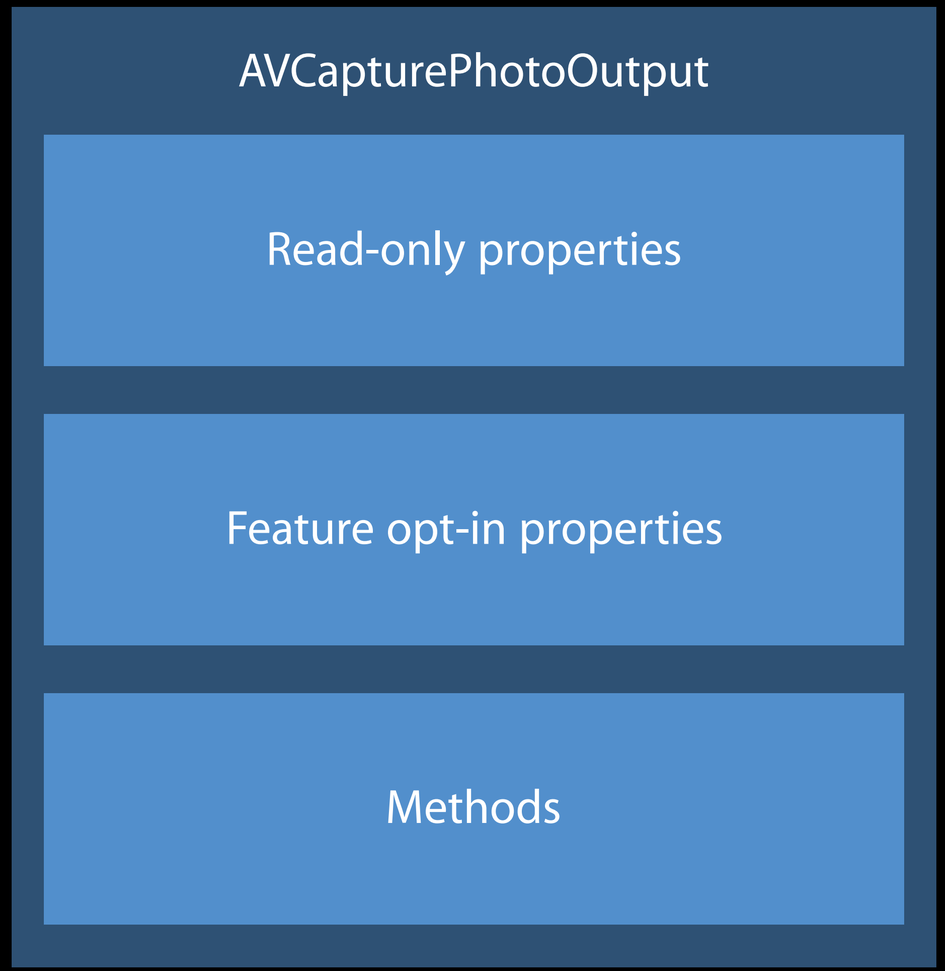
Photo settings encapsulation

A delegate-style interface for tracking the progress of photo capture requests

Resolving of photo settings to an immutable object

Using AVCapturePhotoOutput

NEW



Using AVCapturePhotoOutput

NEW

AVCapturePhotoOutput

isLivePhotoCaptureSupported
availablePhotoPixelFormatTypes
availablePhotoCodecTypes

Feature opt-in properties

Methods

Using AVCapturePhotoOutput

NEW

AVCapturePhotoOutput

isLivePhotoCaptureSupported
availablePhotoPixelFormatTypes
availablePhotoCodecTypes

isHighResolutionCaptureEnabled
isLivePhotoCaptureEnabled

Methods

Using AVCapturePhotoOutput

NEW

AVCapturePhotoOutput

isLivePhotoCaptureSupported
availablePhotoPixelFormatTypes
availablePhotoCodecTypes

isHighResolutionCaptureEnabled
isLivePhotoCaptureEnabled

capturePhoto(with:, delegate:)

Using AVCapturePhotoOutput

NEW

AVCapturePhotoOutput

Using AVCapturePhotoOutput

NEW

AVCapturePhotoOutput

AVCapturePhotoSettings

FeatureX

FeatureY

FeatureZ

Using AVCapturePhotoOutput

NEW

AVCapturePhotoOutput

-capturePhoto(with:,

AVCapturePhotoSettings

FeatureX

FeatureY

FeatureZ

Using AVCapturePhotoOutput

NEW

AVCapturePhotoOutput

-capturePhoto(with:,

AVCapturePhotoSettings

FeatureX

FeatureY

FeatureZ

AVCapturePhotoCaptureDelegate

photoEventAHappened

photoEventBHappened

photoEventCHappened

Using AVCapturePhotoOutput

NEW

AVCapturePhotoOutput

-capturePhoto(with:,

delegate:)

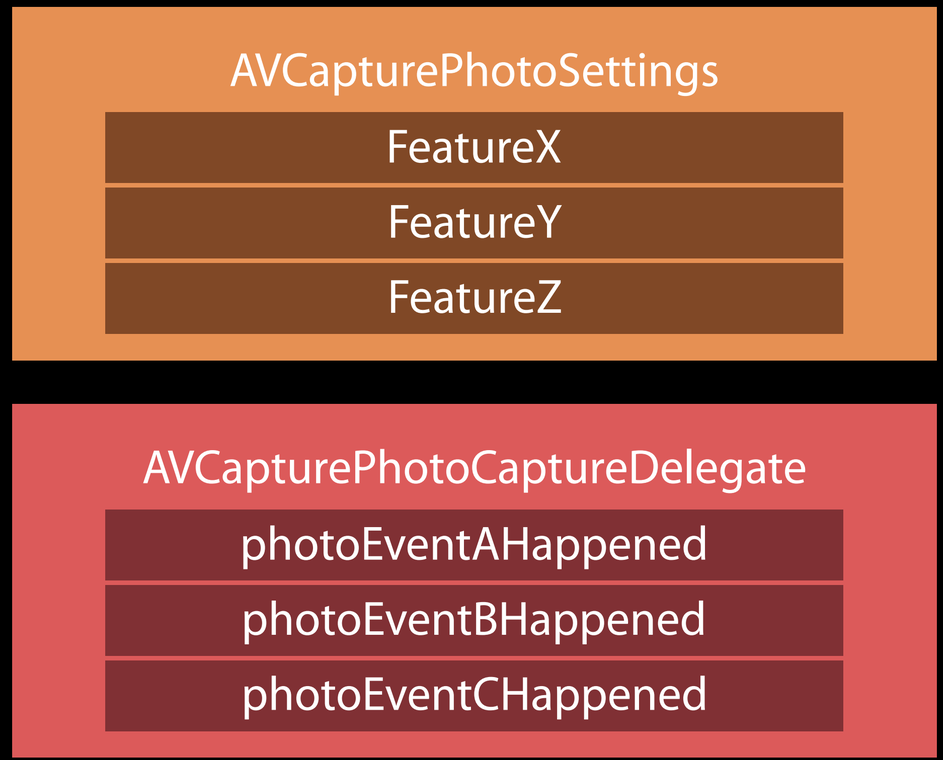
AVCapturePhotoSettings

- FeatureX
- FeatureY
- FeatureZ

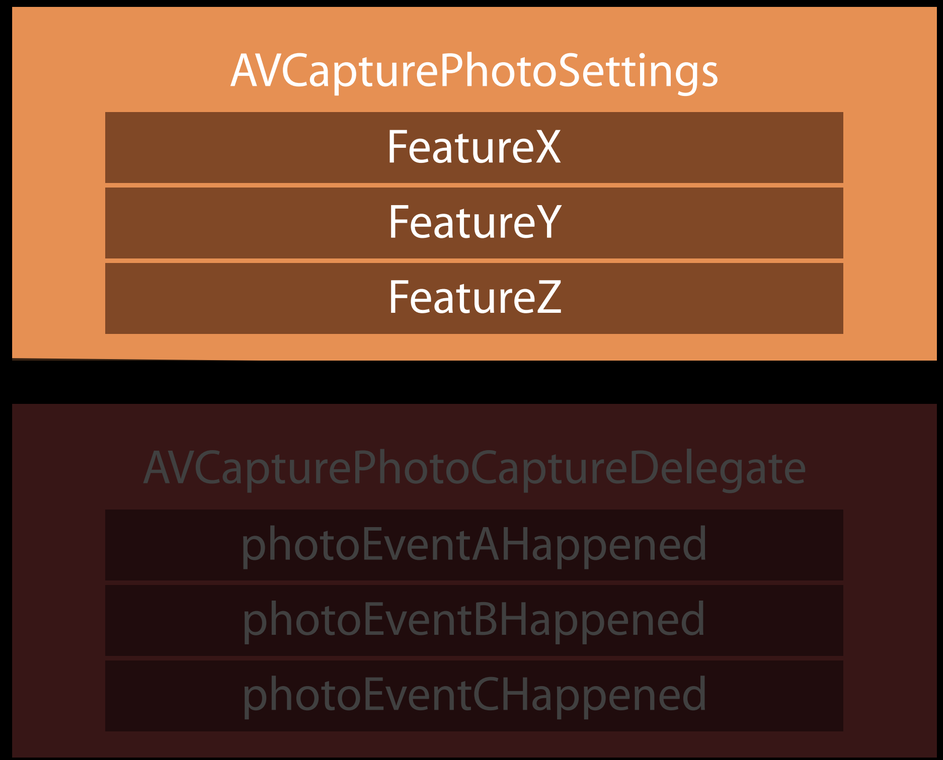
AVCapturePhotoCaptureDelegate

- photoEventAHappened
- photoEventBHappened
- photoEventCHappened

AVCapturePhotoSettings

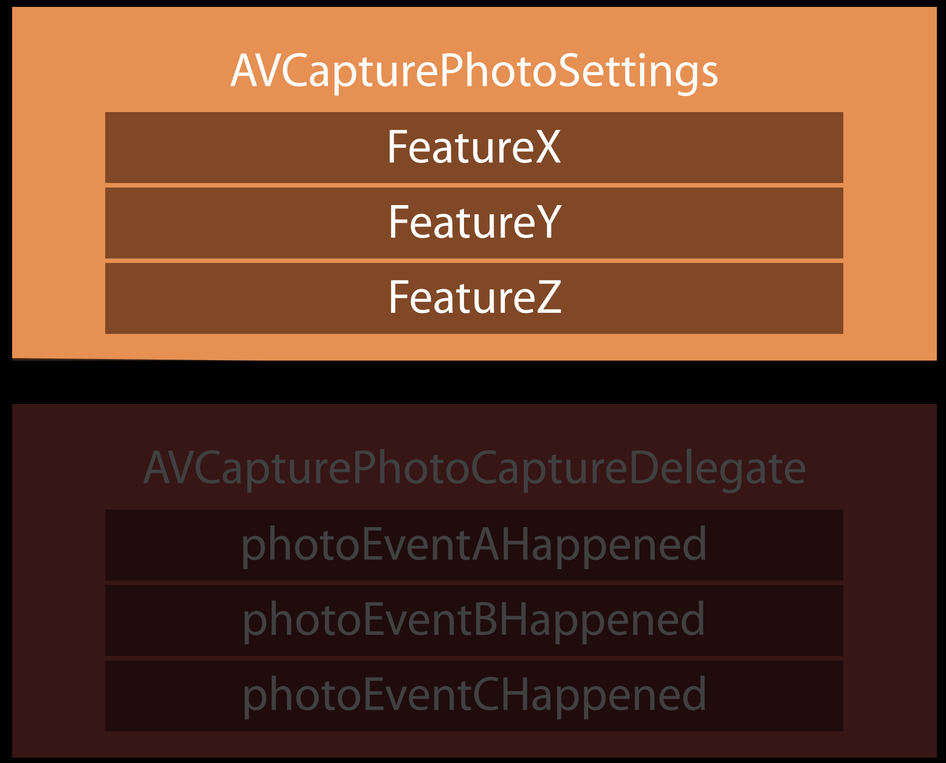


AVCapturePhotoSettings



AVCapturePhotoSettings

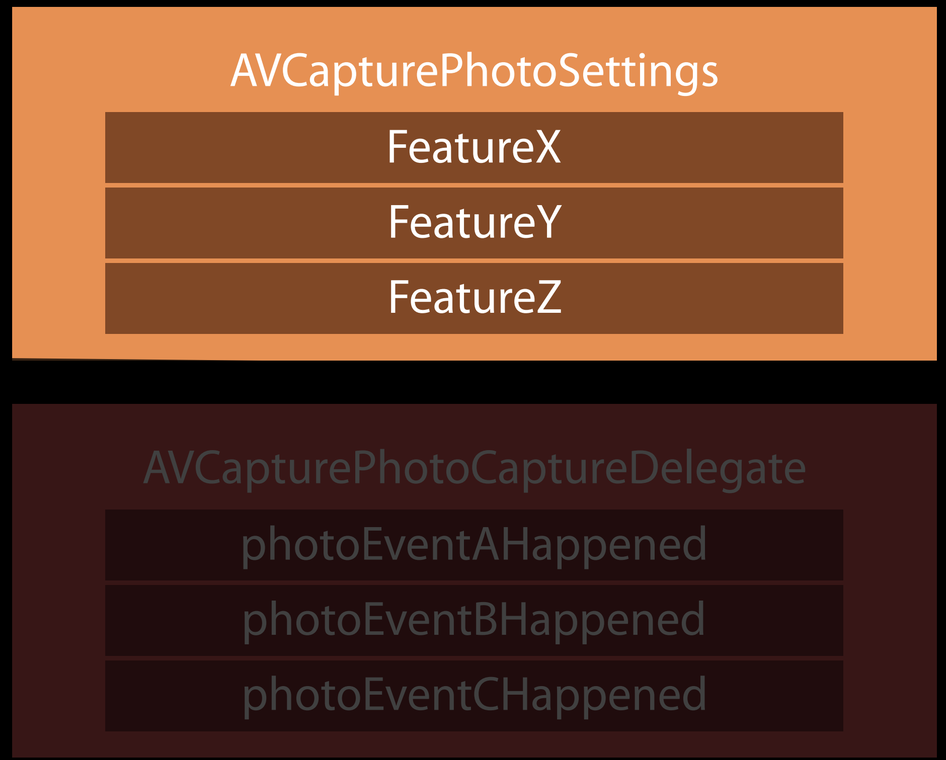
Atomic



AVCapturePhotoSettings

Atomic

Unique

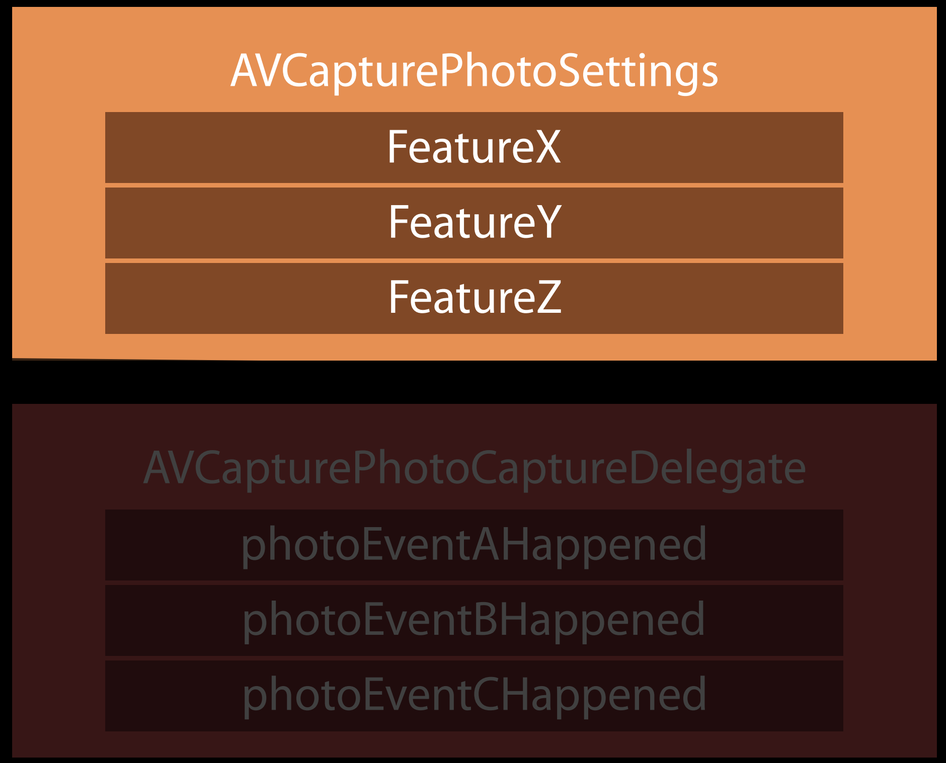


AVCapturePhotoSettings

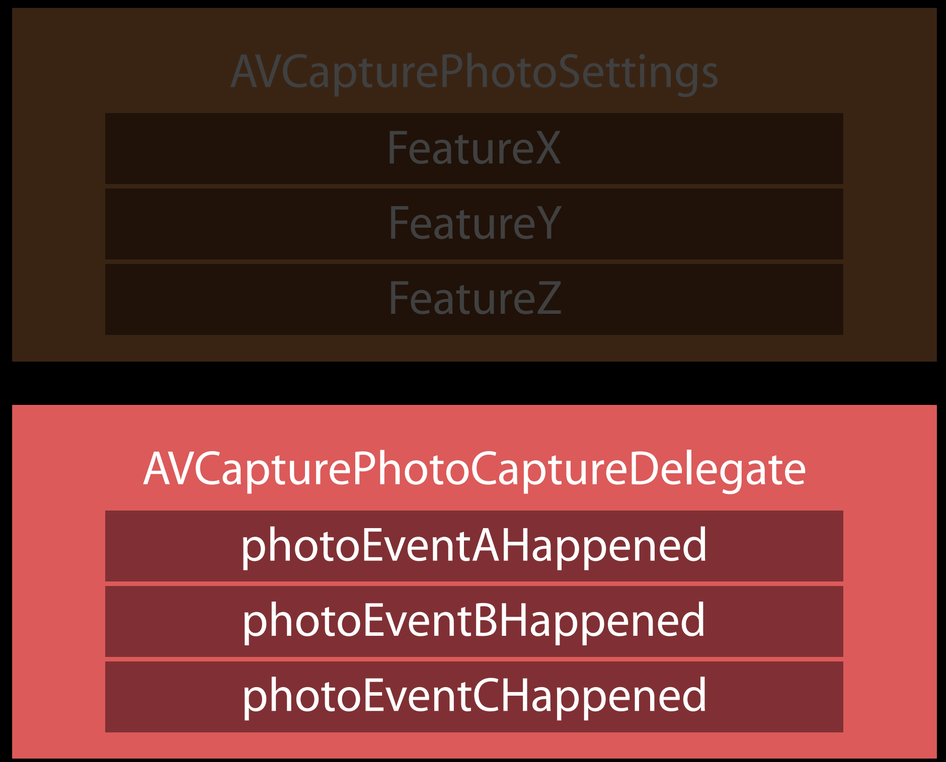
Atomic

Unique

Your order form copy

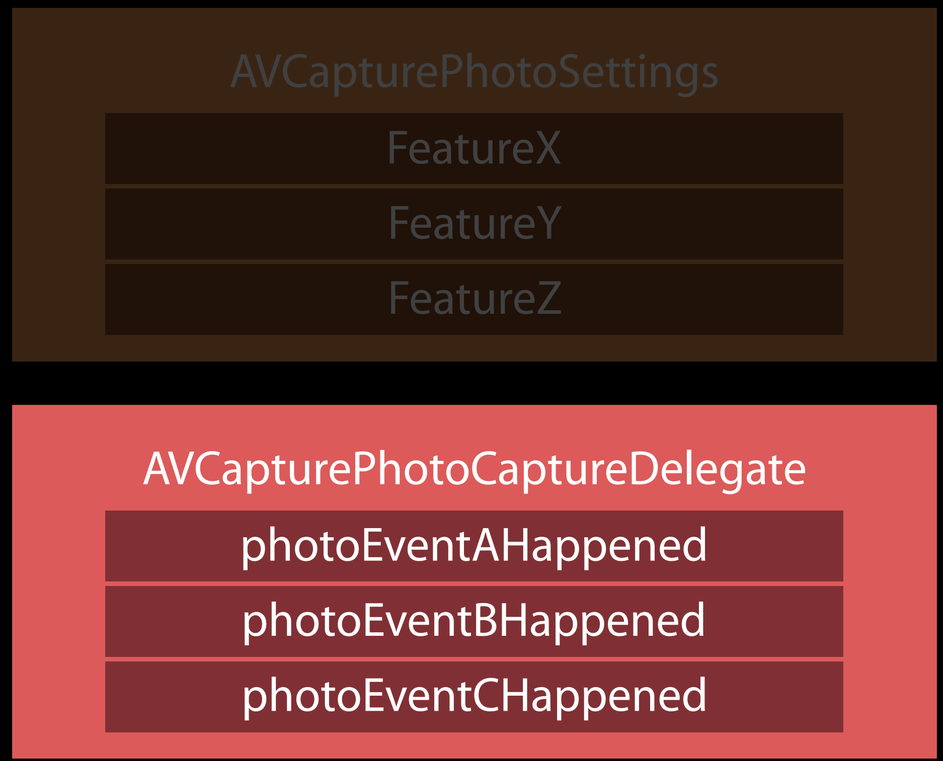


AVCapturePhotoSettings

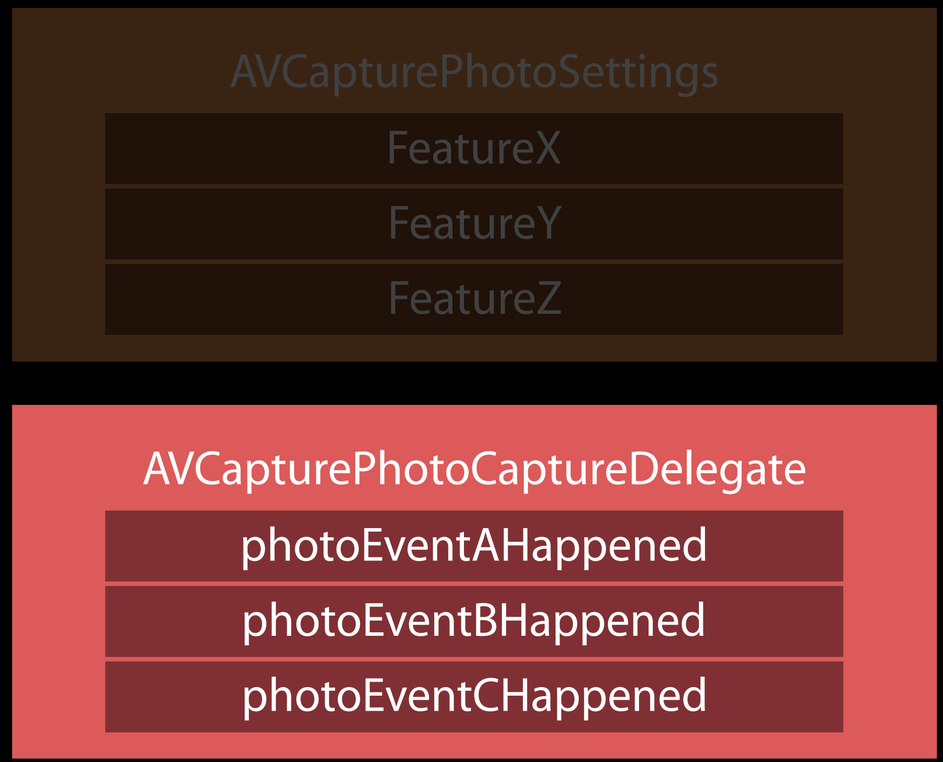


AVCapturePhotoSettings

Single set of callbacks per photo settings



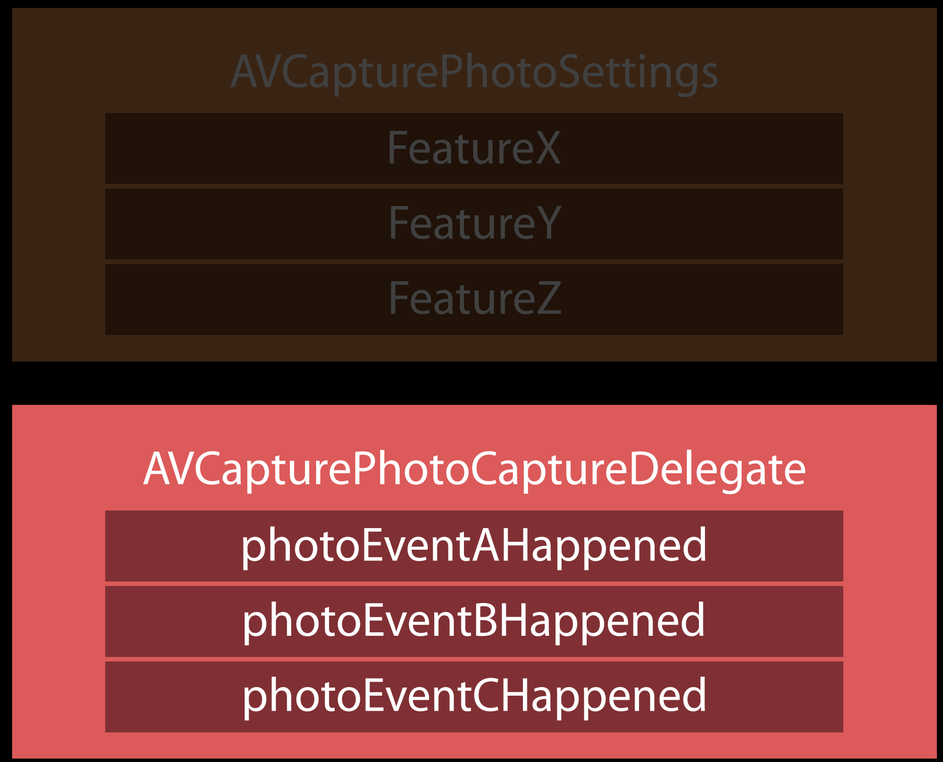
AVCapturePhotoSettings



Single set of callbacks per photo settings

Ordering is documented

AVCapturePhotoSettings

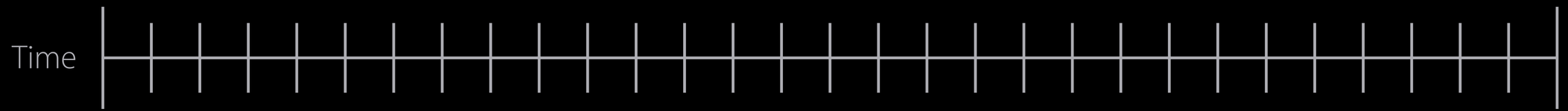


Single set of callbacks per photo settings

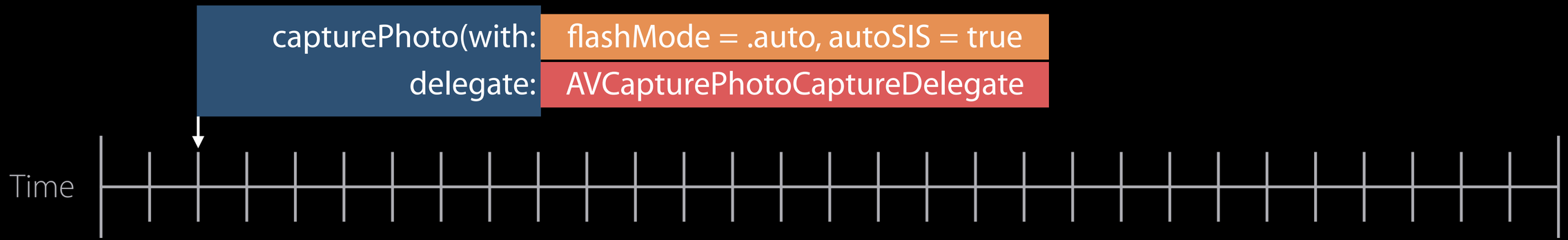
Ordering is documented

Vehicle for resolving indeterminate settings

AVCapturePhotoCaptureDelegate Usage

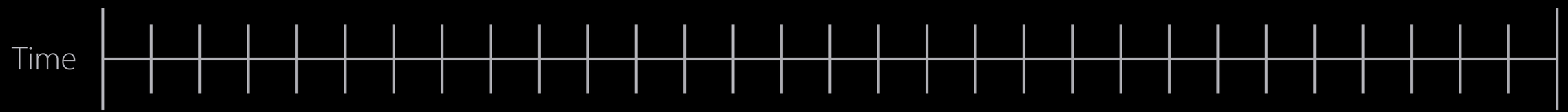


AVCapturePhotoCaptureDelegate Usage



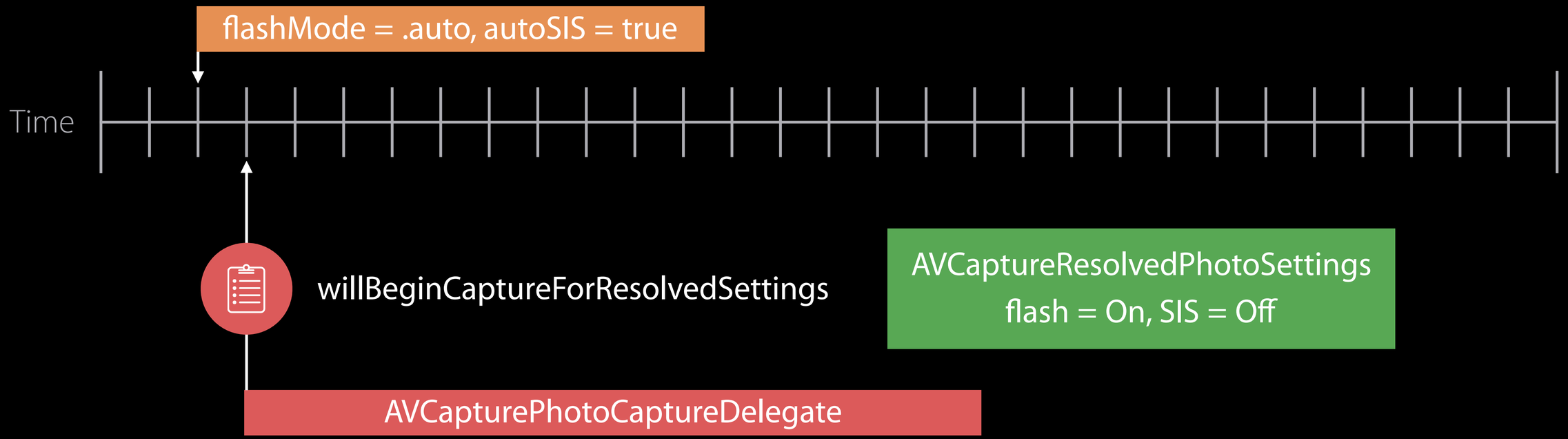
AVCapturePhotoCaptureDelegate Usage

flashMode = .auto, autoSIS = true

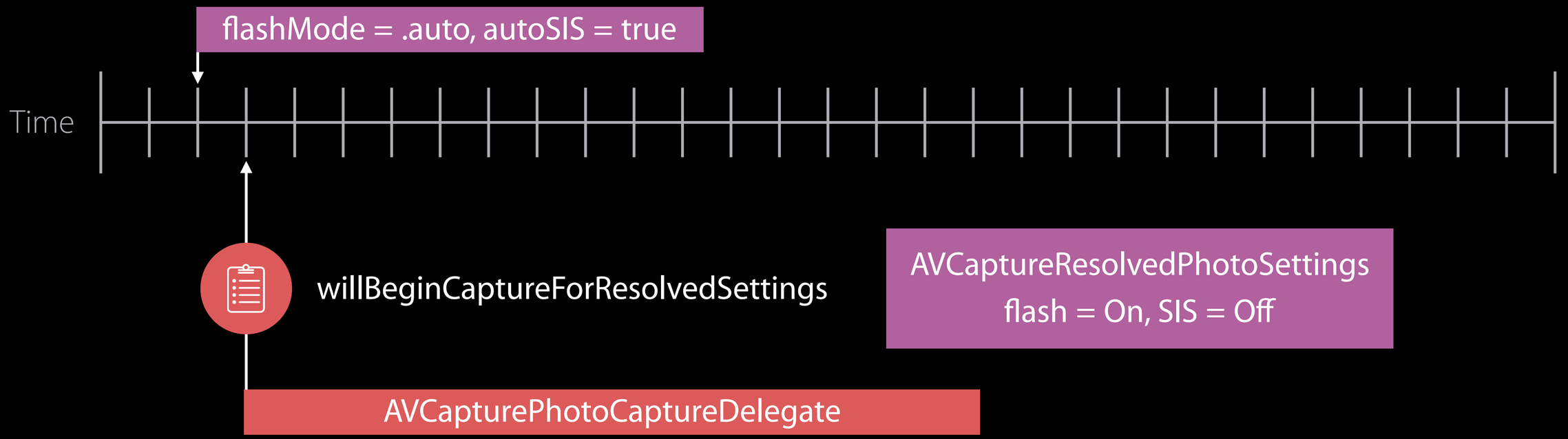


AVCapturePhotoCaptureDelegate

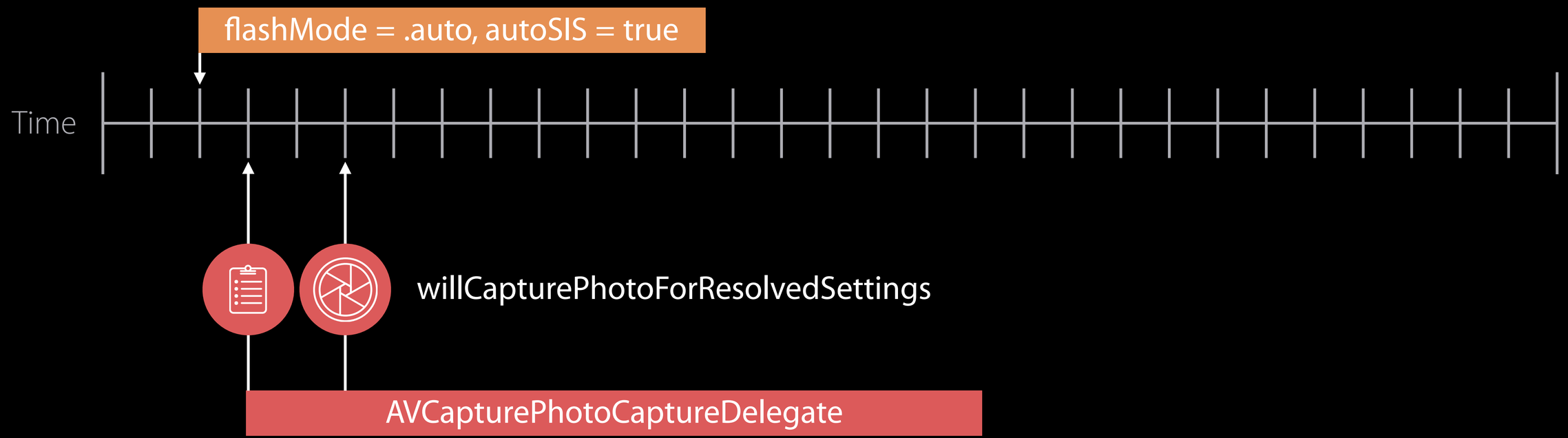
AVCapturePhotoCaptureDelegate Usage



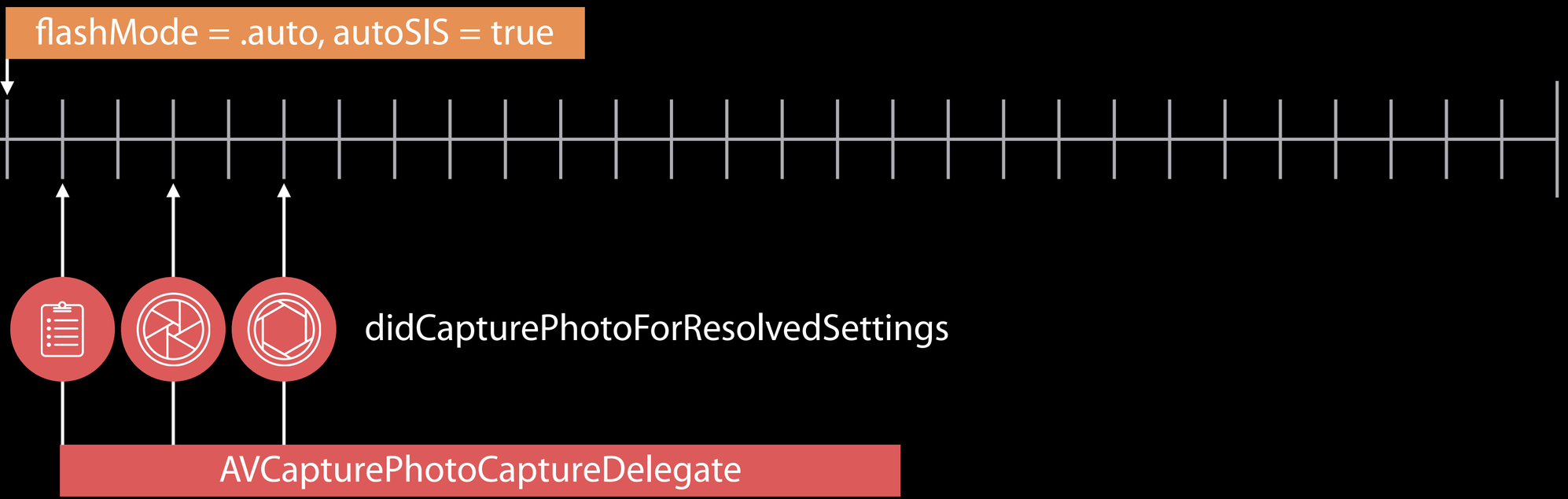
AVCapturePhotoCaptureDelegate Usage



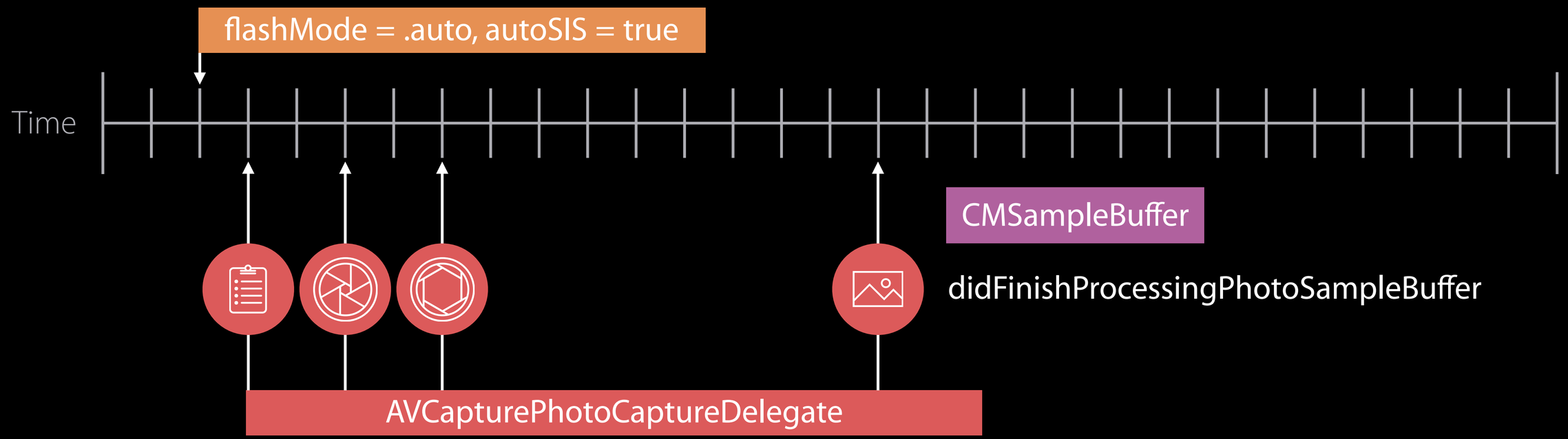
AVCapturePhotoCaptureDelegate Usage



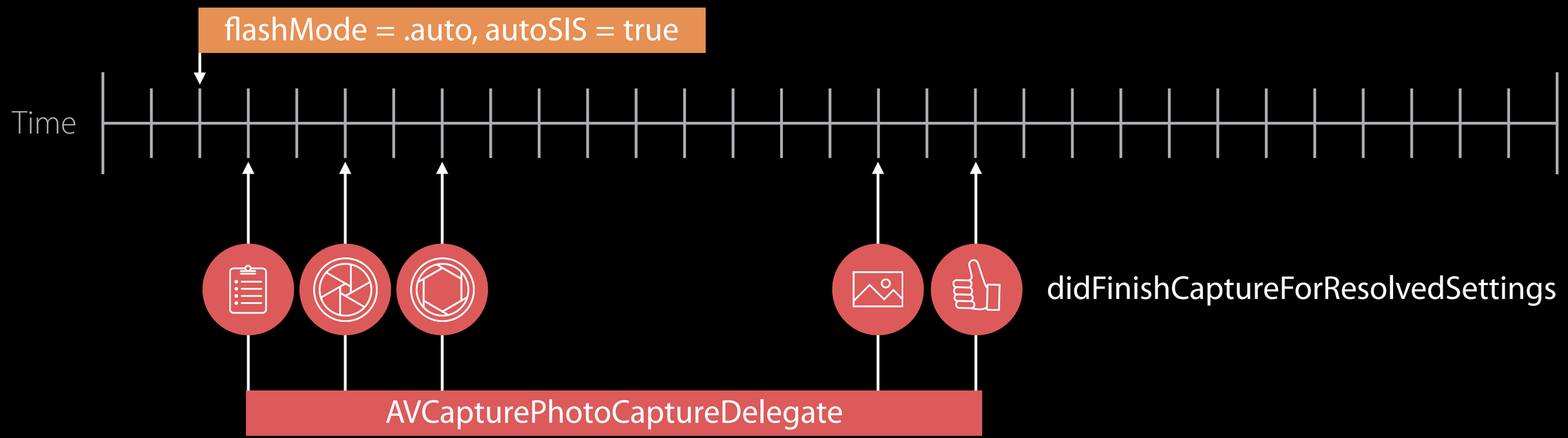
AVCapturePhotoCaptureDelegate Usage



AVCapturePhotoCaptureDelegate Usage



AVCapturePhotoCaptureDelegate Usage



AVCapturePhotoCaptureDelegate Specifics

AVCapturePhotoCaptureDelegate Specifics

Delegate callbacks track a single photo capture request

AVCapturePhotoCaptureDelegate Specifics

Delegate callbacks track a single photo capture request

`AVCapturePhotoOutput` holds a weak reference to your delegate

AVCapturePhotoCaptureDelegate Specifics

Delegate callbacks track a single photo capture request

`AVCapturePhotoOutput` holds a weak reference to your delegate

All callbacks are marked @optional

AVCapturePhotoCaptureDelegate Specifics

Delegate callbacks track a single photo capture request

`AVCapturePhotoOutput` holds a weak reference to your delegate

All callbacks are marked @optional

Some callbacks are required at runtime depending on your photo settings

AVCapturePhotoCaptureDelegate Specifics

Delegate callbacks track a single photo capture request

`AVCapturePhotoOutput` holds a weak reference to your delegate

All callbacks are marked @optional

Some callbacks are required at runtime depending on your photo settings

All callbacks pass an instance of `AVCaptureResolvedPhotoSettings`

```swift
// Initiating a photo capture using AVCapturePhotoOutput

func takeHighResolutionPhoto()
{
   let settings = AVCapturePhotoSettings()
   settings.isHighResolutionPhotoEnabled = true

   photoOutput.capturePhoto(with: settings, delegate: self)
}

func takeFlashPhoto()
{
   let settings = AVCapturePhotoSettings()
   settings.flashMode = .auto

   photoOutput.capturePhoto(with: settings, delegate: self)
}

func takeBGRAPhoto()
{
    let bgraFormat: [String : AnyObject] = [kCVPixelBufferPixelFormatTypeKey as String :
                                            NSNumber(value: kCVPixelFormatType_32BGRA)]
    let settings = AVCapturePhotoSettings(format: bgraFormat)

    photoOutput.capturePhoto(with: settings, delegate: self)
}
```

```
// Initiating a photo capture using AVCapturePhotoOutput

func takeHighResolutionPhoto()
{
   let settings = AVCapturePhotoSettings()
   settings.isHighResolutionPhotoEnabled = true

   photoOutput.capturePhoto(with: settings, delegate: self)
}

func takeFlashPhoto()
{
   let settings = AVCapturePhotoSettings()
   settings.flashMode = .auto

   photoOutput.capturePhoto(with: settings, delegate: self)
}

func takeBGRAPhoto()
{
    let bgraFormat: [String : AnyObject] = [kCVPixelBufferPixelFormatTypeKey as String :
                                            NSNumber(value: kCVPixelFormatType_32BGRA)]
    let settings = AVCapturePhotoSettings(format: bgraFormat)

    photoOutput.capturePhoto(with: settings, delegate: self)
}
```

```swift
// Initiating a photo capture using AVCapturePhotoOutput

func takeHighResolutionPhoto()
{
   let settings = AVCapturePhotoSettings()
   settings.isHighResolutionPhotoEnabled = true

   photoOutput.capturePhoto(with: settings, delegate: self)
}

func takeFlashPhoto()
{
   let settings = AVCapturePhotoSettings()
   settings.flashMode = .auto

   photoOutput.capturePhoto(with: settings, delegate: self)
}

func takeBGRAPhoto()
{
    let bgraFormat: [String : AnyObject] = [kCVPixelBufferPixelFormatTypeKey as String :
                                            NSNumber(value: kCVPixelFormatType_32BGRA)]
    let settings = AVCapturePhotoSettings(format: bgraFormat)

    photoOutput.capturePhoto(with: settings, delegate: self)
}
```

```swift
// Initiating a photo capture using AVCapturePhotoOutput

func takeHighResolutionPhoto()
{
   let settings = AVCapturePhotoSettings()
   settings.isHighResolutionPhotoEnabled = true

   photoOutput.capturePhoto(with: settings, delegate: self)
}

func takeFlashPhoto()
{
   let settings = AVCapturePhotoSettings()
   settings.flashMode = .auto

   photoOutput.capturePhoto(with: settings, delegate: self)
}
```

```swift
func takeBGRAPhoto()
{
    let bgraFormat: [String : AnyObject] = [kCVPixelBufferPixelFormatTypeKey as String :
                                            NSNumber(value: kCVPixelFormatType_32BGRA)]
    let settings = AVCapturePhotoSettings(format: bgraFormat)

    photoOutput.capturePhoto(with: settings, delegate: self)
}
```

AVCaptureResolvedPhotoSettings Properties

AVCaptureResolvedPhotoSettings Properties

```
public var uniqueID: Int64 { get }
```

AVCaptureResolvedPhotoSettings Properties

```
public var uniqueID: Int64 { get }
public var photoDimensions: CMVideoDimensions { get }
```

AVCaptureResolvedPhotoSettings Properties

```
public var uniqueID: Int64 { get }
public var photoDimensions: CMVideoDimensions { get }
public var isFlashEnabled: Bool { get }
```

AVCaptureResolvedPhotoSettings Properties

```
public var uniqueID: Int64 { get }
public var photoDimensions: CMVideoDimensions { get }
public var isFlashEnabled: Bool { get }
public var isStillImageStabilizationEnabled: Bool { get }
```

Bracketed Capture Support

Bracketed Capture Support

Review 2014 Session 508: Camera Capture Manual Controls

Bracketed Capture Support

Review 2014 Session 508: Camera Capture Manual Controls

Auto Exposure and Manual Exposure Brackets

Bracketed Capture Support

Review 2014 Session 508: Camera Capture Manual Controls

Auto Exposure and Manual Exposure Brackets

AVCapturePhotoBracketSettings is a `subclass` of AVCapturePhotoSettings

Bracketed Capture Support

Review 2014 Session 508: Camera Capture Manual Controls

Auto Exposure and Manual Exposure Brackets

AVCapturePhotoBracketSettings is a `subclass` of AVCapturePhotoSettings

```
public var bracketedSettings: [AVCaptureBracketedStillImageSettings] { get }
public var isLensStabilizationEnabled: Bool
```

Bracketed Capture Support

Review 2014 Session 508: Camera Capture Manual Controls

Auto Exposure and Manual Exposure Brackets

AVCapturePhotoBracketSettings is a `subclass` of AVCapturePhotoSettings

```
public var bracketedSettings: [AVCaptureBracketedStillImageSettings] { get }
public var isLensStabilizationEnabled: Bool
```

```
func capture(_ captureOutput: AVCapturePhotoOutput,
             didFinishProcessingPhotoSampleBuffer photoSampleBuffer: CMSampleBuffer?,
             previewPhotoSampleBuffer: CMSampleBuffer?,
             resolvedSettings: AVCaptureResolvedPhotoSettings,
             bracketSettings: AVCaptureBracketedStillImageSettings?,
             error: NSError?)
{
    // bracketSettings tells you which item in your array of bracketedSettings
    // this image corresponds to.
}
```

Bracketed Capture Support

Review 2014 Session 508: Camera Capture Manual Controls

Auto Exposure and Manual Exposure Brackets

AVCapturePhotoBracketSettings is a `subclass` of AVCapturePhotoSettings

```
public var bracketedSettings: [AVCaptureBracketedStillImageSettings] { get }
public var isLensStabilizationEnabled: Bool
```

```
func capture(_ captureOutput: AVCapturePhotoOutput,
             didFinishProcessingPhotoSampleBuffer photoSampleBuffer: CMSampleBuffer?,
             previewPhotoSampleBuffer: CMSampleBuffer?,
             resolvedSettings: AVCaptureResolvedPhotoSettings,
             bracketSettings: AVCaptureBracketedStillImageSettings?,
             error: NSError?)
{
    // bracketSettings tells you which item in your array of bracketedSettings
    // this image corresponds to.
}
```

Deprecated in iOS 10

- AVCaptureStillImageOutput
- AVCaptureDevice.flashActive
- AVCaptureDevice isFlashModeSupported(_:)
- AVCaptureDevice.flashMode

Deprecated in iOS 10

✗ AVCaptureStillImageOutput	✓ AVCapturePhotoOutput
✗ AVCaptureDevice.flashActive	✓ AVCapturePhotoOutput.isFlashScene
✗ AVCaptureDevice isFlashModeSupported(_:)	✓ AVCapturePhotoOutput.supportedFlashModes
✗ AVCaptureDevice.flashMode	✓ AVCapturePhotoSettings.flashMode

AVCapturePhotoOutput Benefits

AVCapturePhotoOutput Benefits

Easier bookkeeping

AVCapturePhotoOutput Benefits

Easier bookkeeping

Immediate settings resolution

AVCapturePhotoOutput Benefits

Easier bookkeeping

Immediate settings resolution

Confident request tracking

AVCapturePhotoOutput Benefits

Easier bookkeeping

Immediate settings resolution

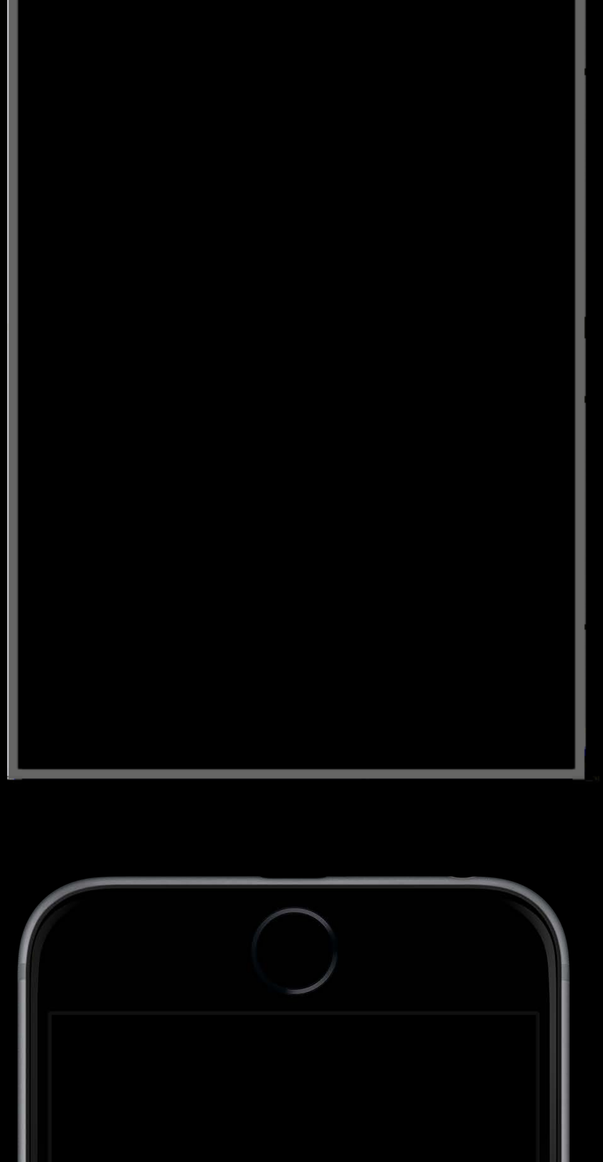
Confident request tracking

Expandable palette of callbacks

Live Photos

Capturing memories

“A still photo captures an instant frozen in time.
With Live Photos, you can turn those instants into unforgettable living memories.”



What is a Live Photo

Moment or memory?

What is a Live Photo

Moment or memory?

It’s a moment!

What is a Live Photo

Moment or memory?

It’s a moment!

- Full resolution still image (12 MP JPEG)

What is a Live Photo

Moment or memory?

It’s a moment!

- Full resolution still image (12 MP JPEG)
- Same quality as non-Live Photo (SIS / OIS)

What is a Live Photo

Moment or memory?

It's a moment!

- Full resolution still image (12 MP JPEG)
- Same quality as non-Live Photo (SIS / OIS)
- "Frictionless" capture

What is a Live Photo

Moment or memory?

What is a Live Photo

Moment or memory?

It’s a memory!

What is a Live Photo

Moment or memory?

It’s a memory!

- A short movie encompassing the still time

What is a Live Photo

Moment or memory?

It's a memory!

- A short movie encompassing the still time
- Screen resolution (1440x1080 or 1290x960)

What is a Live Photo

Moment or memory?

It's a memory!

- A short movie encompassing the still time
- Screen resolution (1440x1080 or 1290x960)
- Includes audio

What is a Live Photo

Moment or memory?

It's a memory!

- A short movie encompassing the still time
- Screen resolution (1440x1080 or 1290x960)
- Includes audio

In iOS 9.1 — "Shoe shot" auto trimming

What is a Live Photo

Moment or memory?

It's a memory!

- A short movie encompassing the still time
- Screen resolution (1440x1080 or 1290x960)
- Includes audio

In iOS 9.1 — "Shoe shot" auto trimming

New in iOS 10 — Video stabilization

What is a Live Photo

Moment or memory?

It's a memory!

- A short movie encompassing the still time
- Screen resolution (1440x1080 or 1290x960)
- Includes audio

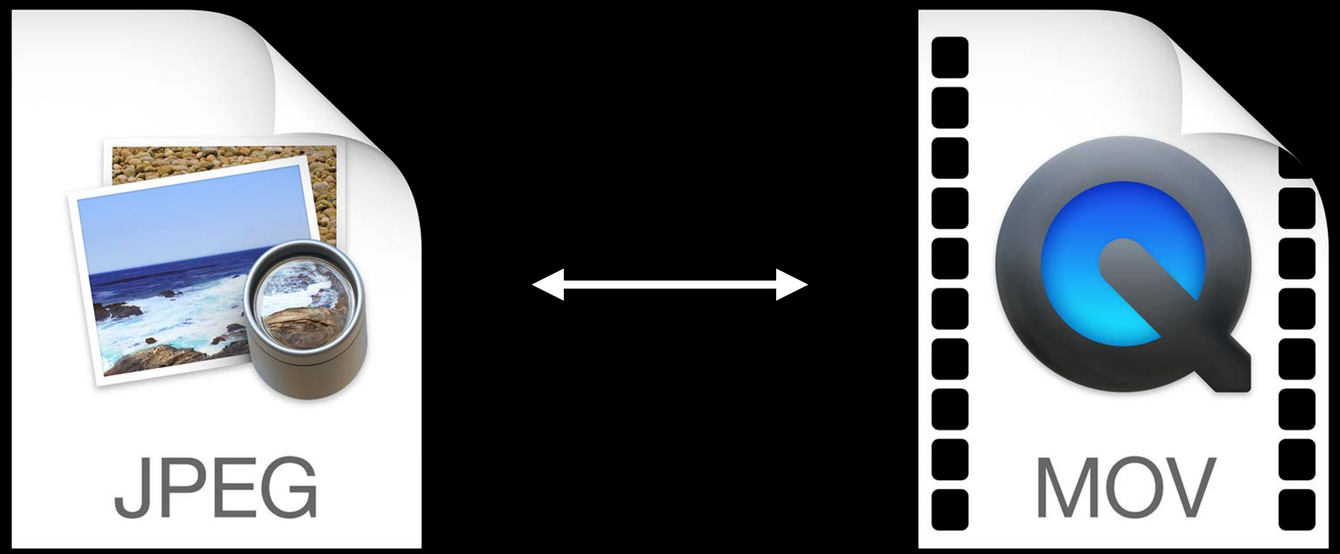
In iOS 9.1 — "Shoe shot" auto trimming

New in iOS 10 — Video stabilization

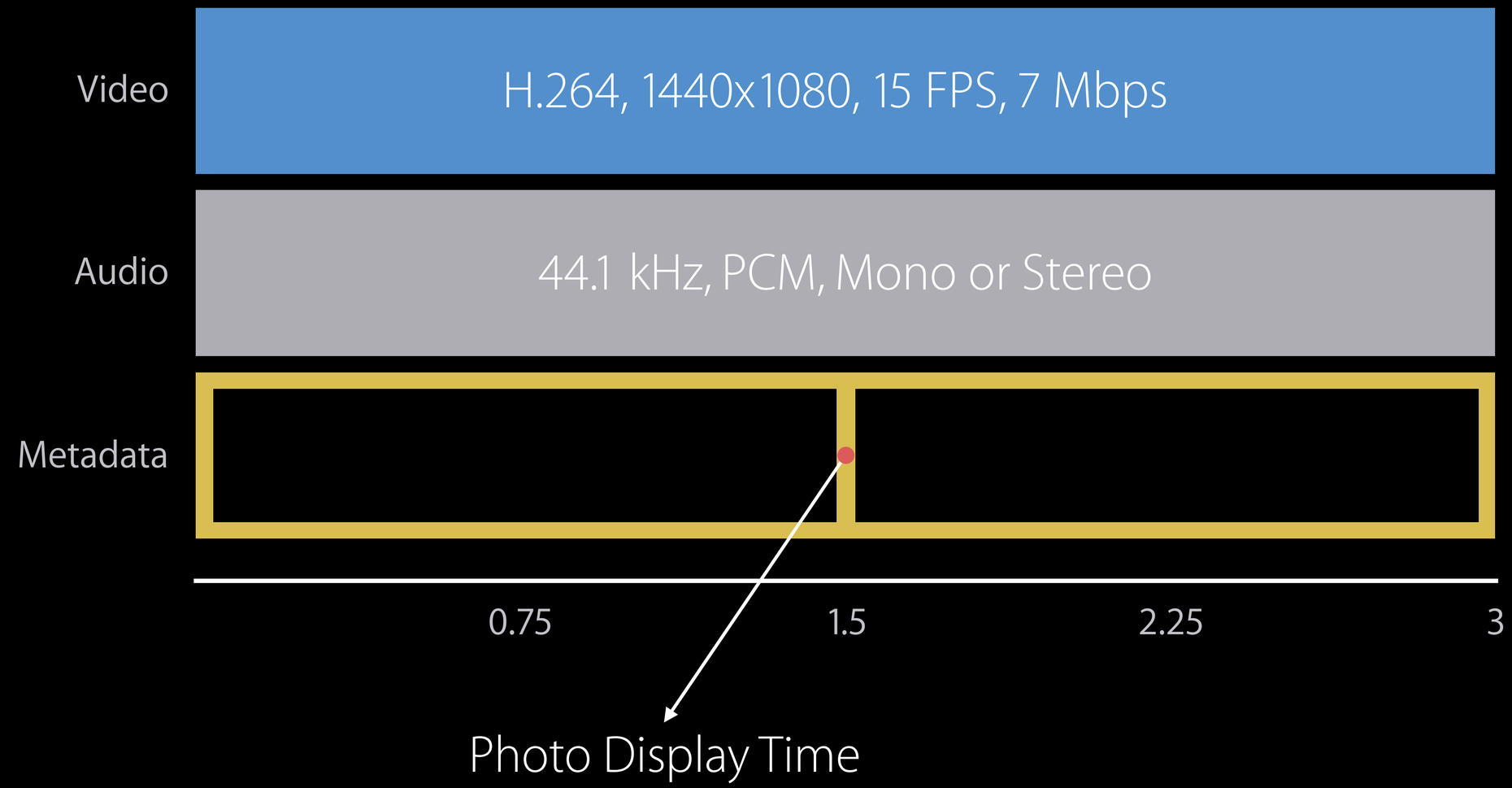
New in iOS 10 — Interruption-free music during capture

What is a Live Photo

It's a moment and a memory



Live Photo .MOV Asset



Capturing Live Photos

Capturing Live Photos

Check `AVCapturePhotoOutput.isLivePhotoCaptureSupported`

Capturing Live Photos

Check `AVCapturePhotoOutput.isLivePhotoCaptureSupported`

Live Photo capture is only supported with the `AVCaptureSessionPresetPhoto`

Capturing Live Photos

Check `AVCapturePhotoOutput.isLivePhotoCaptureSupported`

Live Photo capture is only supported with the `AVCaptureSessionPresetPhoto`

Opt in using `AVCapturePhotoOutput.isLivePhotoCaptureEnabled = true`

Capturing Live Photos

Check `AVCapturePhotoOutput.isLivePhotoCaptureSupported`

Live Photo capture is only supported with the `AVCaptureSessionPresetPhoto`

Opt in using `AVCapturePhotoOutput.isLivePhotoCaptureEnabled = true`

Add an AVCaptureDeviceInput for the microphone to your AVCaptureSession

Capturing Live Photos

Check `AVCapturePhotoOutput.isLivePhotoCaptureSupported`

Live Photo capture is only supported with the `AVCaptureSessionPresetPhoto`

Opt in using `AVCapturePhotoOutput.isLivePhotoCaptureEnabled = true`

Add an AVCaptureDeviceInput for the microphone to your AVCaptureSession

Presence of AVCaptureMovieFileOutput disables Live Photo capture

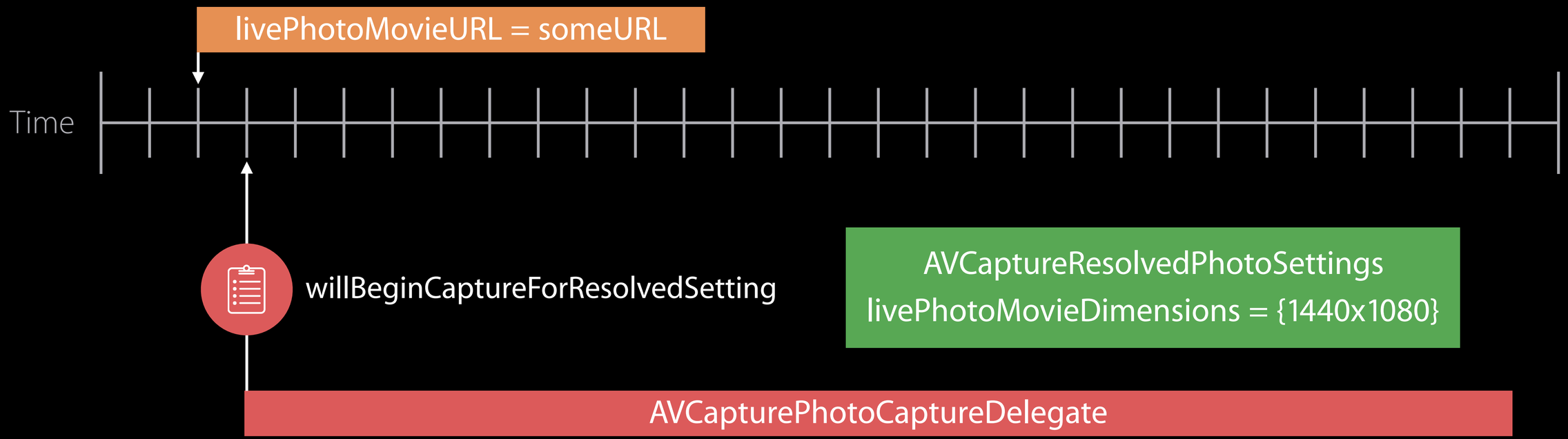
```swift
// Live Photo Capture Example

func takeLivePhoto()
{
    let settings = AVCapturePhotoSettings()
    settings.isHighResolutionPhotoEnabled = true

    settings.livePhotoMovieFileURL = URL(fileURLWithPath:
                    "\(NSTemporaryDirectory())/\(_uniqueAppSignature)_\(settings.uniqueID)")

     // Optional Movie-level Metadata
     var metadataItems = [AVMetadataItem]()

     let authorMetadataItem = AVMutableMetadataItem()
     authorMetadataItem.keySpace = AVMetadataKeySpaceCommon
     authorMetadataItem.key = AVMetadataCommonKeyAuthor
     authorMetadataItem.value = "Brad Ford"
     metadataItems.append(authorMetadataItem)

     settings.livePhotoMovieMetadata = metadataItems

     photoOutput.capturePhoto(with: settings, delegate: self)
}
```

```
// Live Photo Capture Example

func takeLivePhoto()
{
    let settings = AVCapturePhotoSettings()
    settings.isHighResolutionPhotoEnabled = true

    settings.livePhotoMovieFileURL = URL(fileURLWithPath:
                    "\(NSTemporaryDirectory())/\(_uniqueAppSignature)_\(settings.uniqueID)")

     // Optional Movie-level Metadata
     var metadataItems = [AVMetadataItem]()

     let authorMetadataItem = AVMutableMetadataItem()
     authorMetadataItem.keySpace = AVMetadataKeySpaceCommon
     authorMetadataItem.key = AVMetadataCommonKeyAuthor
     authorMetadataItem.value = "Brad Ford"
     metadataItems.append(authorMetadataItem)

     settings.livePhotoMovieMetadata = metadataItems

     photoOutput.capturePhoto(with: settings, delegate: self)
}
```

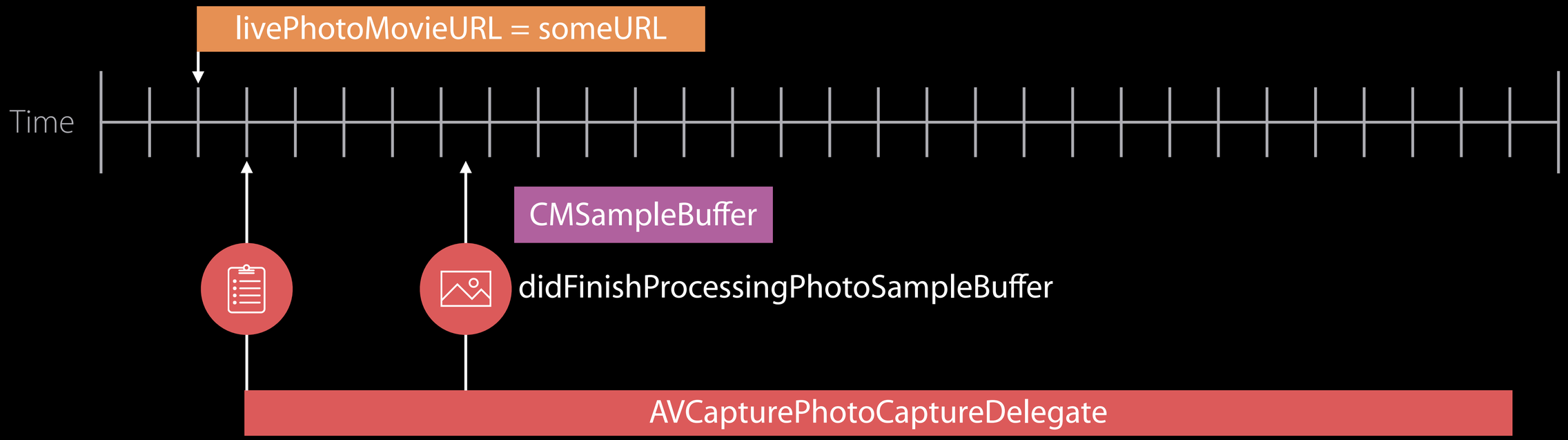
```
// Live Photo Capture Example

func takeLivePhoto()
{
    let settings = AVCapturePhotoSettings()
    settings.isHighResolutionPhotoEnabled = true

    settings.livePhotoMovieFileURL = URL(fileURLWithPath:
                        "\(NSTemporaryDirectory())/\(_uniqueAppSignature)_\(settings.uniqueID)")

    // Optional Movie-level Metadata
    var metadataItems = [AVMetadataItem]()

    let authorMetadataItem = AVMutableMetadataItem()
    authorMetadataItem.keySpace = AVMetadataKeySpaceCommon
    authorMetadataItem.key = AVMetadataCommonKeyAuthor
    authorMetadataItem.value = "Brad Ford"
    metadataItems.append(authorMetadataItem)

    settings.livePhotoMovieMetadata = metadataItems

    photoOutput.capturePhoto(with: settings, delegate: self)
}
```

```
// Live Photo Capture Example

func takeLivePhoto()
{
    let settings = AVCapturePhotoSettings()
    settings.isHighResolutionPhotoEnabled = true

    settings.livePhotoMovieFileURL = URL(fileURLWithPath:
                    "\(NSTemporaryDirectory())/\(_uniqueAppSignature)_\(settings.uniqueID)")

     // Optional Movie-level Metadata
     var metadataItems = [AVMetadataItem]()

     let authorMetadataItem = AVMutableMetadataItem()
     authorMetadataItem.keySpace = AVMetadataKeySpaceCommon
     authorMetadataItem.key = AVMetadataCommonKeyAuthor
     authorMetadataItem.value = "Brad Ford"
     metadataItems.append(authorMetadataItem)

     settings.livePhotoMovieMetadata = metadataItems

    photoOutput.capturePhoto(with: settings, delegate: self)
}
```

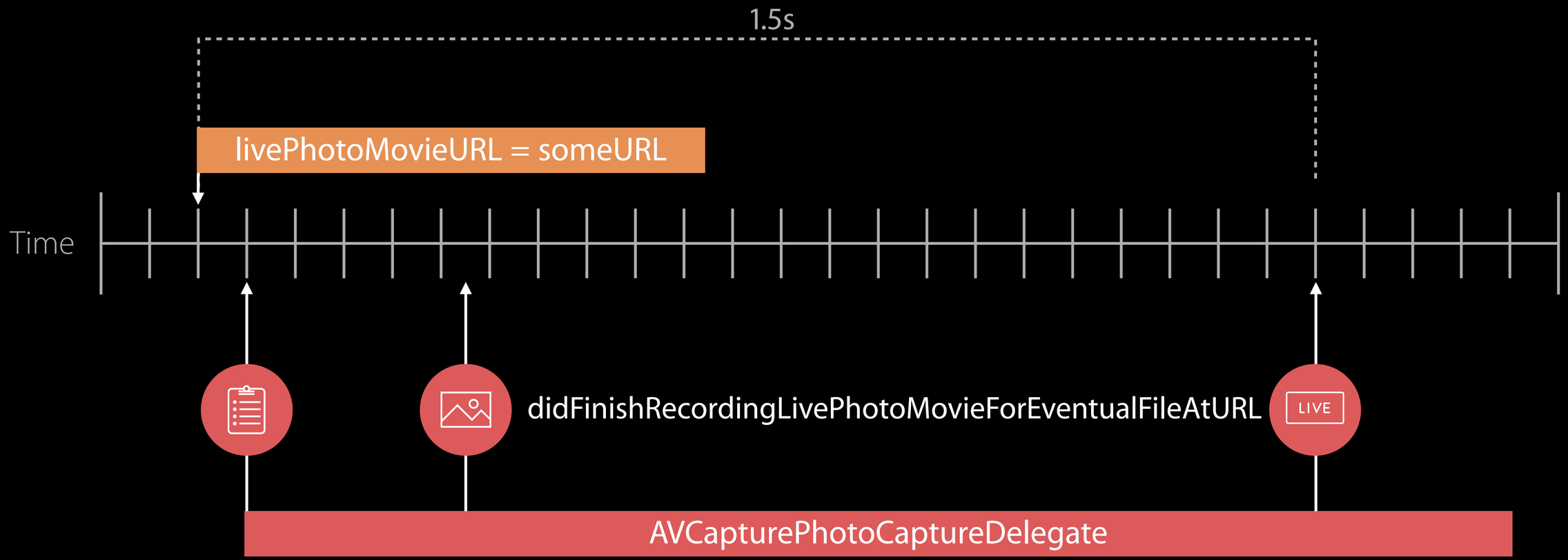
Live Photo Delegate Methods



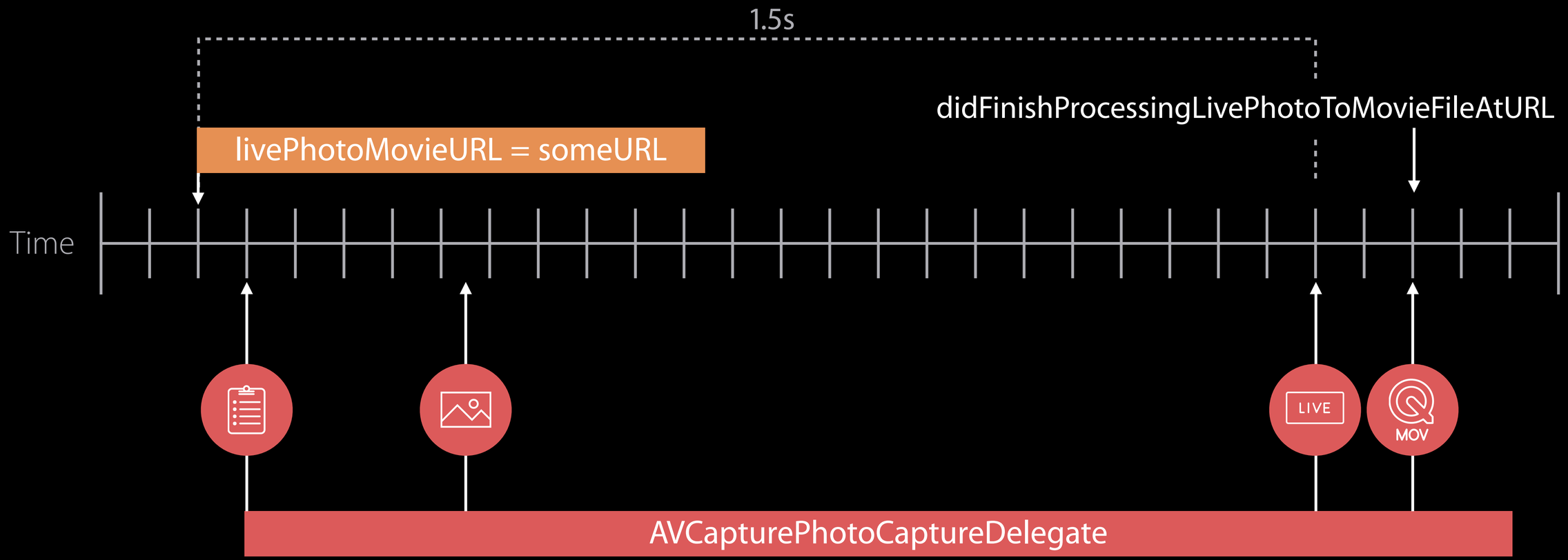
Live Photo Delegate Methods



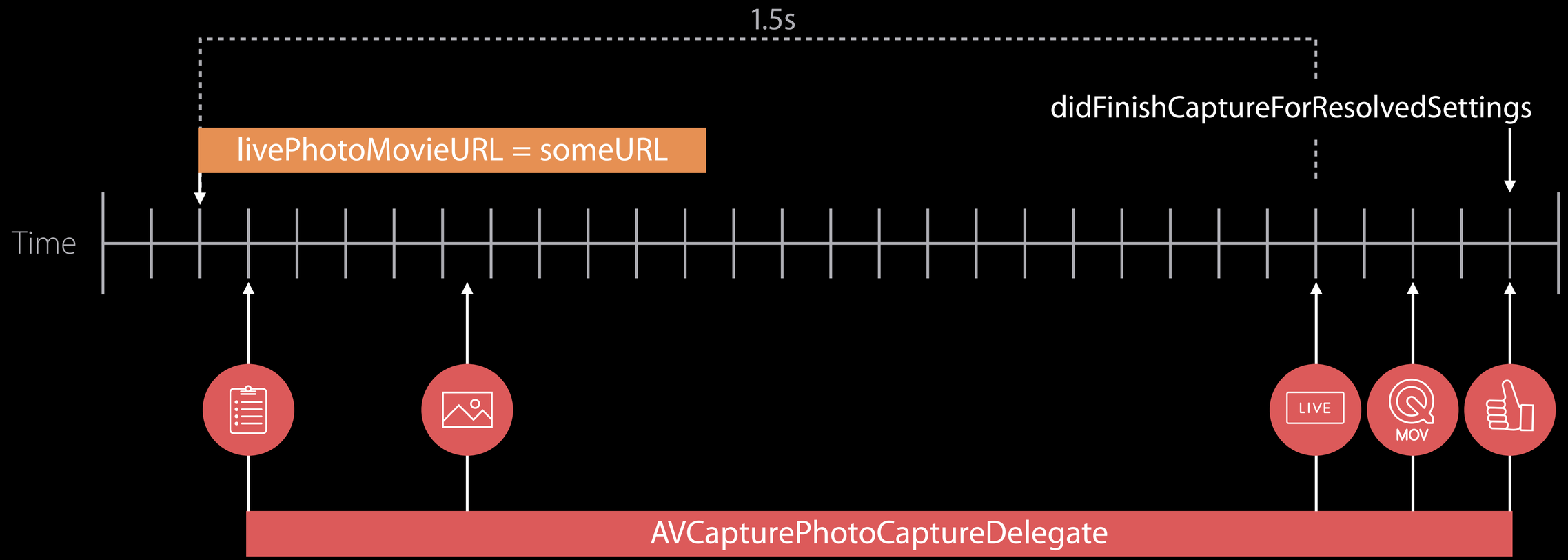
Live Photo Delegate Methods



Live Photo Delegate Methods



Live Photo Delegate Methods



```
// Live Photo Capture Delegate Methods

func capture(_ captureOutput: AVCapturePhotoOutput,
                 didFinishProcessingPhotoSampleBuffer photoSampleBuffer: CMSampleBuffer?,
                 previewPhotoSampleBuffer: CMSampleBuffer?,
                 resolvedSettings: AVCaptureResolvedPhotoSettings,
                 bracketSettings: AVCaptureBracketedStillImageSettings?,
                 error: NSError?)
{
    if let photoSampleBuffer = photoSampleBuffer,
          data = AVCapturePhotoOutput.jpegPhotoDataRepresentation(forJPEGSampleBuffer:
                                  photoSampleBuffer, previewPhotoSampleBuffer: previewPhotoSampleBuffer)
        {
         do {
             try data.write(to: _uniqueJPEGFilePath, options: .atomicWrite)
         }
         catch {
           // Handle error
         }
    }
}
```

```
// Live Photo Capture Delegate Methods

func capture(_ captureOutput: AVCapturePhotoOutput,
                 didFinishProcessingPhotoSampleBuffer photoSampleBuffer: CMSampleBuffer?,
                 previewPhotoSampleBuffer: CMSampleBuffer?,
                 resolvedSettings: AVCaptureResolvedPhotoSettings,
                 bracketSettings: AVCaptureBracketedStillImageSettings?,
                 error: NSError?)
{
    if let photoSampleBuffer = photoSampleBuffer,
          data = AVCapturePhotoOutput.jpegPhotoDataRepresentation(forJPEGSampleBuffer:
                                photoSampleBuffer, previewPhotoSampleBuffer: previewPhotoSampleBuffer)
        {
         do {
             try data.write(to: _uniqueJPEGFilePath, options: .atomicWrite)
         }
         catch {
           // Handle error
         }
    }
}
```

```swift
// Live Photo Capture Delegate Methods

func capture(_ captureOutput: AVCapturePhotoOutput,
                 didFinishProcessingPhotoSampleBuffer photoSampleBuffer: CMSampleBuffer?,
                 previewPhotoSampleBuffer: CMSampleBuffer?,
                 resolvedSettings: AVCaptureResolvedPhotoSettings,
                 bracketSettings: AVCaptureBracketedStillImageSettings?,
                 error: NSError?)
{
    if let photoSampleBuffer = photoSampleBuffer,
          data = AVCapturePhotoOutput.jpegPhotoDataRepresentation(forJPEGSampleBuffer:
                                    photoSampleBuffer, previewPhotoSampleBuffer: previewPhotoSampleBuffer)
        {
         do {
             try data.write(to: _uniqueJPEGFilePath, options: .atomicWrite)
         }
         catch {
           // Handle error
         }
    }
}
```

AVCapturePhotoCaptureDelegate Tip

AVCapturePhotoCaptureDelegate Tip

Some types of photo captures deliver multiple assets

AVCapturePhotoCaptureDelegate Tip

Some types of photo captures deliver multiple assets

Instantiate a new AVCapturePhotoCaptureDelegate object for each photo request

AVCapturePhotoCaptureDelegate Tip

Some types of photo captures deliver multiple assets

Instantiate a new AVCapturePhotoCaptureDelegate object for each photo request

Aggregate assets in the delegate object

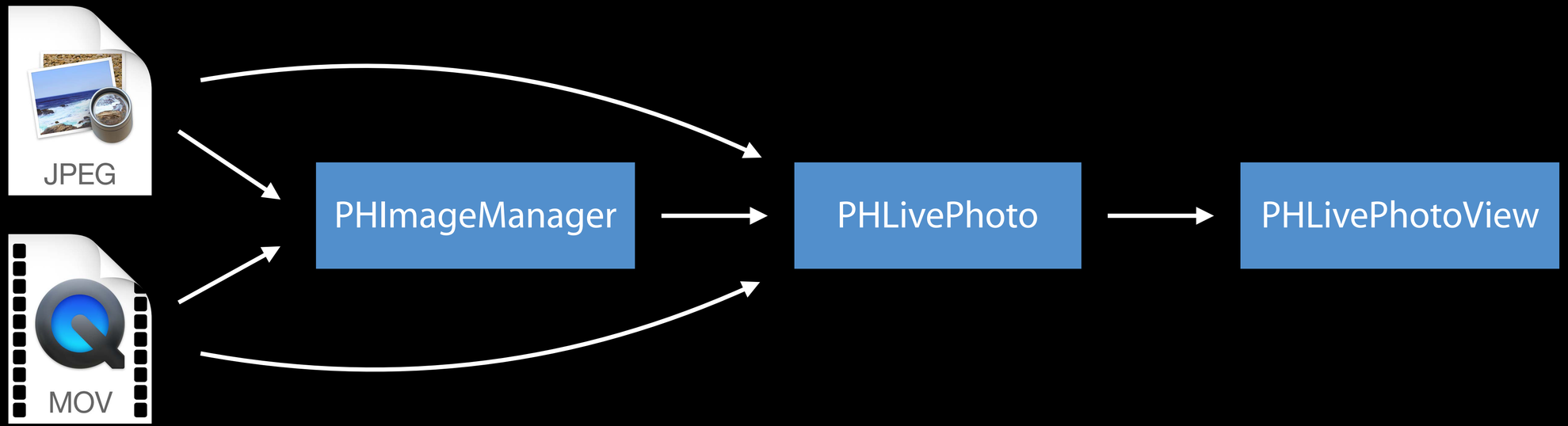
AVCapturePhotoCaptureDelegate Tip

Some types of photo captures deliver multiple assets

Instantiate a new AVCapturePhotoCaptureDelegate object for each photo request

Aggregate assets in the delegate object

Dispose of the object after didFinishCaptureForResolvedSettings:


JPEG
MOV
PHImageManager
PHLivePhoto
PHLivePhotoView

PHLivePhotoEditingContext

NEW

PHLivePhotoEditingContext

Demo

Live Photo Capture and Editing

AVCam and LivePhotoEditor

Live Photo Capture in AVCam

Live Photo Capture in AVCam

Separate video recording and photo modes

Live Photo Capture in AVCam

Separate video recording and photo modes

Shows proper “Live” badging technique

Live Photo Capture in AVCam

Separate video recording and photo modes

Shows proper “Live” badging technique

Saves assets to Photo Library

Live Photo Capture in AVCam

Separate video recording and photo modes

Shows proper “Live” badging technique

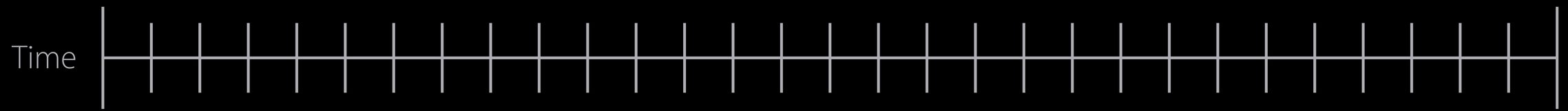
Saves assets to Photo Library

Sample code available now!

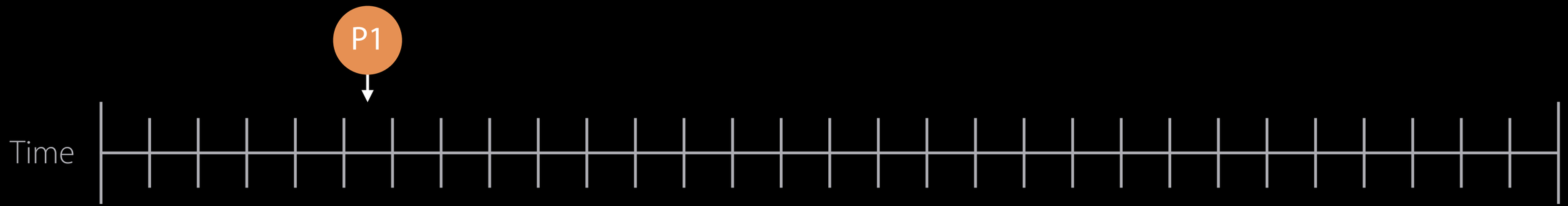
More Live Photo Editing in Session 505

Live Photo Editing and RAW Processing with Core Image	Nob Hill	Thursday 11:00AM

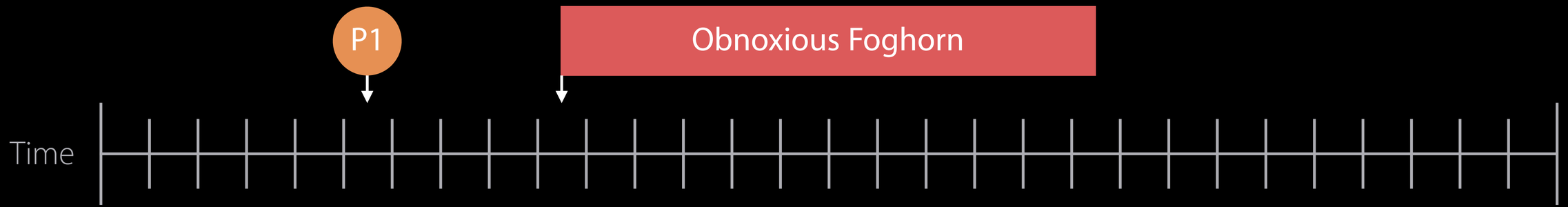
Live Photo Capture Suspension



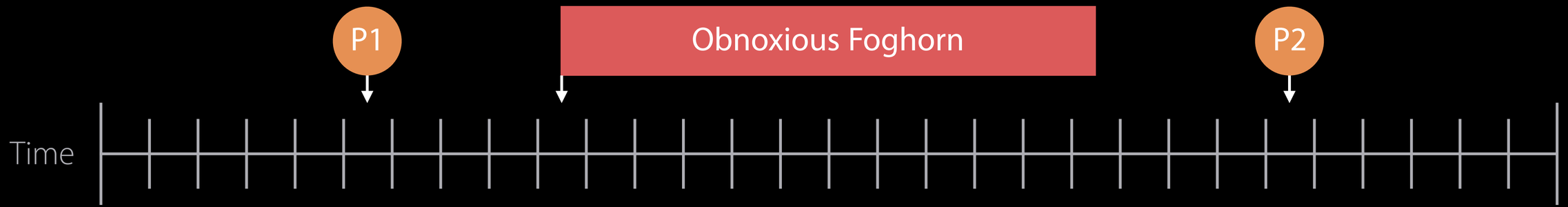
Live Photo Capture Suspension



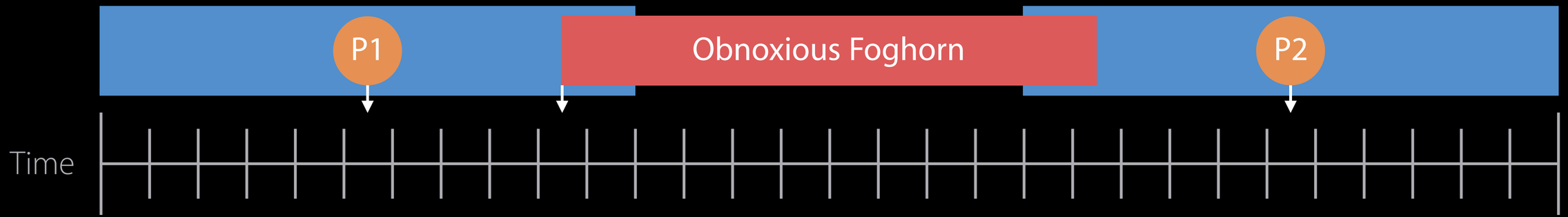
Live Photo Capture Suspension



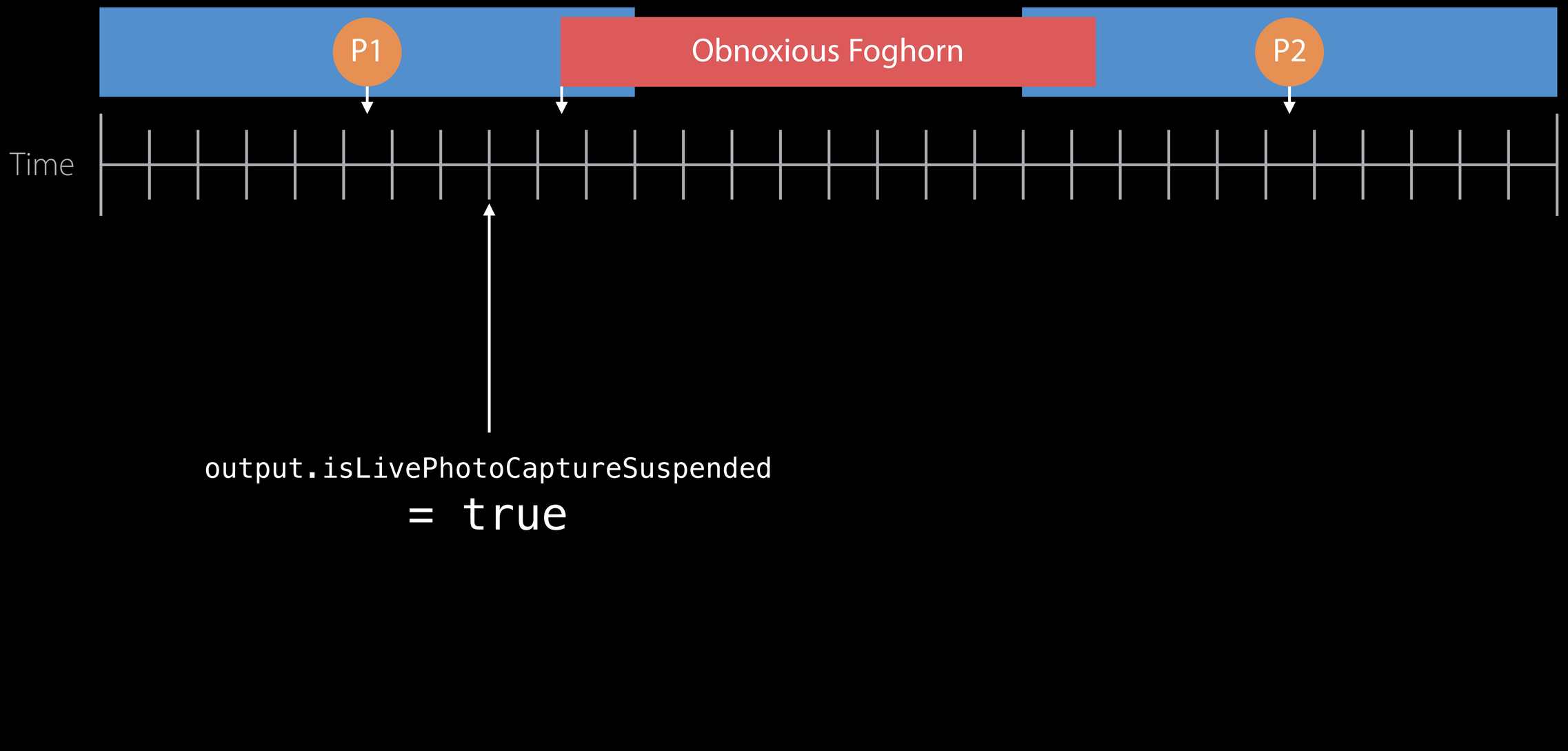
Live Photo Capture Suspension



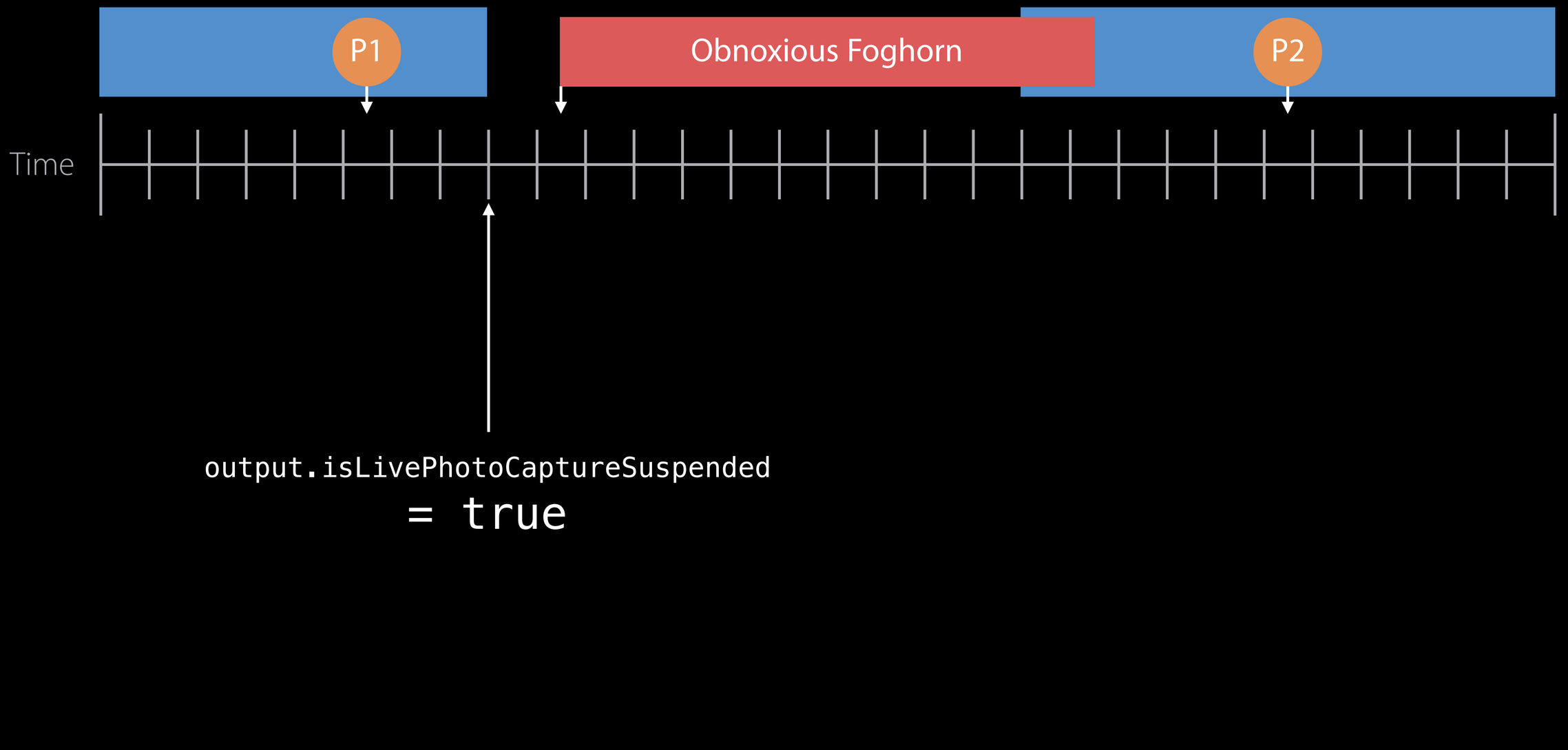
Live Photo Capture Suspension



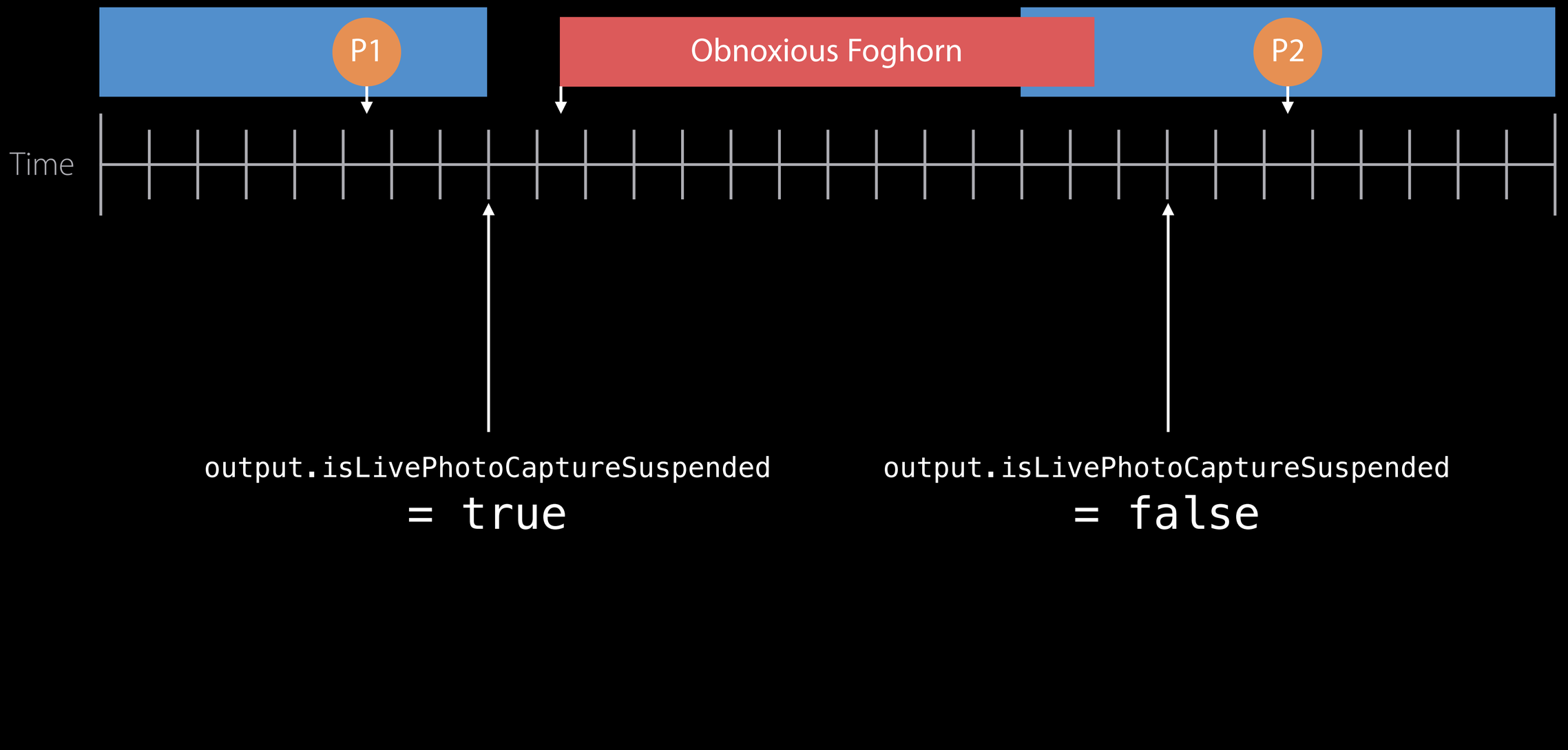
Live Photo Capture Suspension



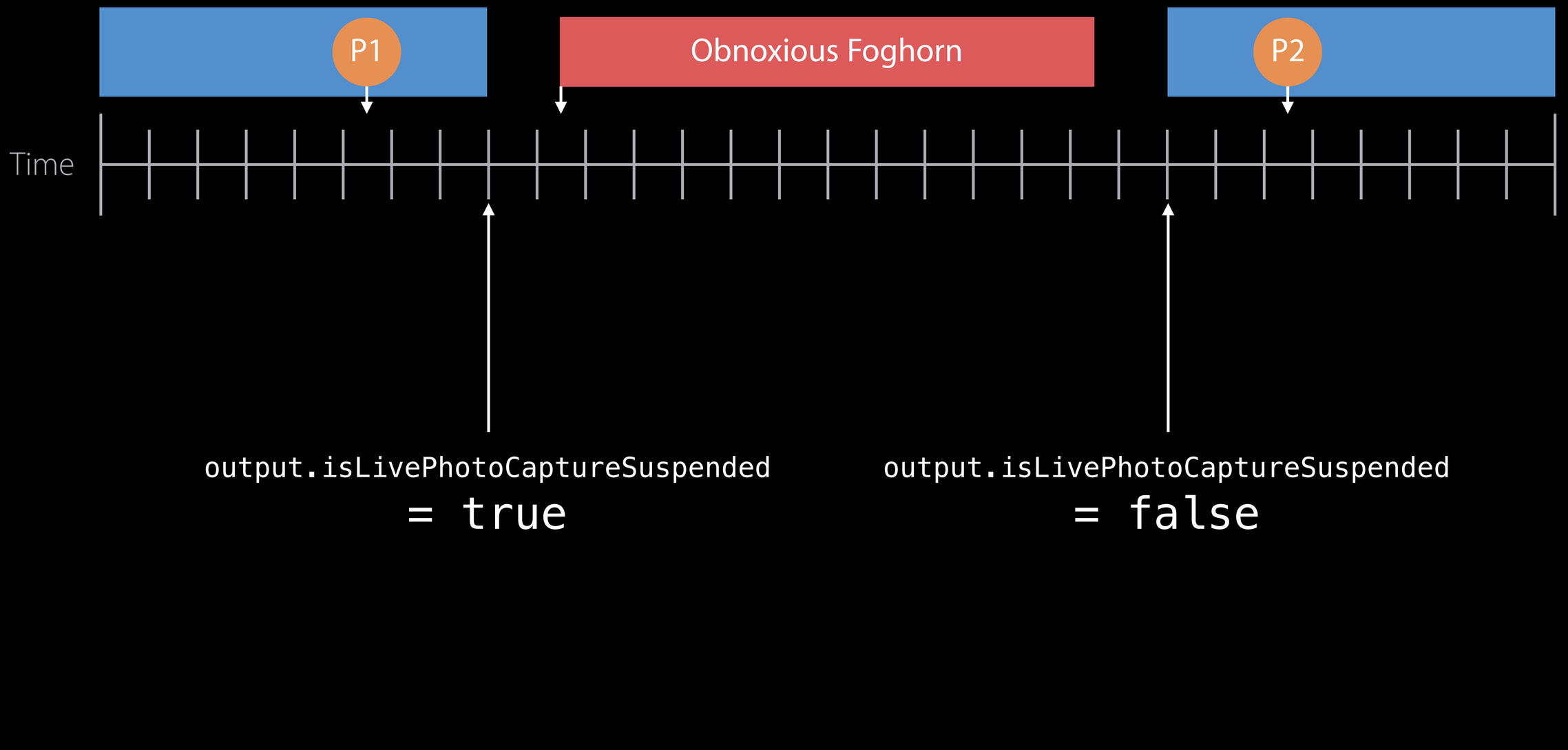
Live Photo Capture Suspension



Live Photo Capture Suspension



Live Photo Capture Suspension



Live Photo Capture Support

Live Photo Capture Support

iPhone 6s

iPhone 6s Plus

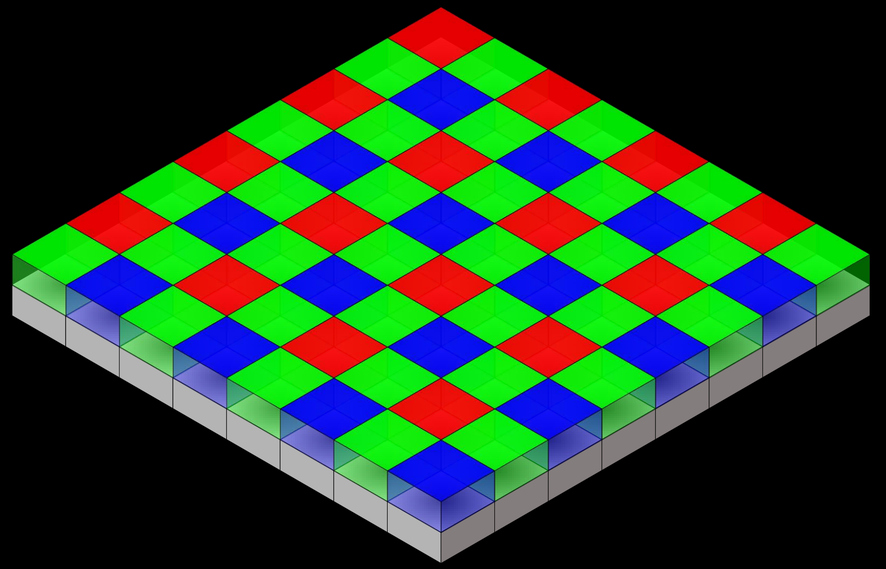
iPhone SE

9.7-inch iPad Pro

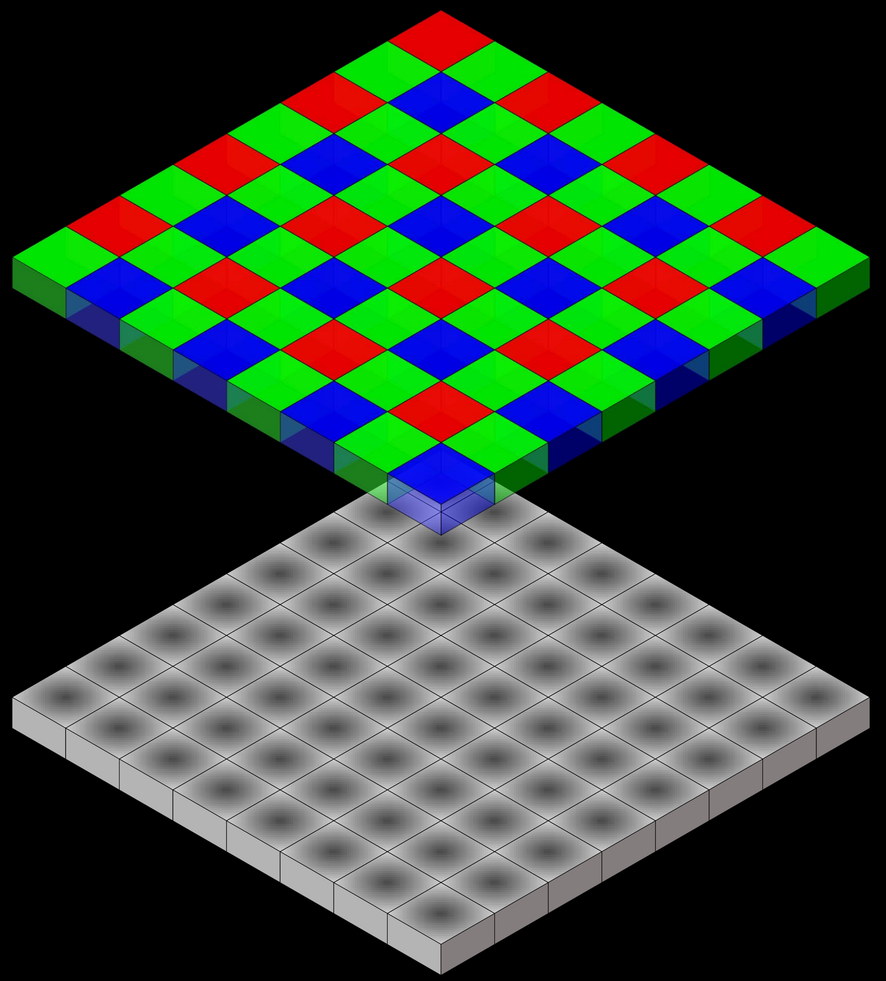
RAW Photo Capture

with a dash of DNG on the side

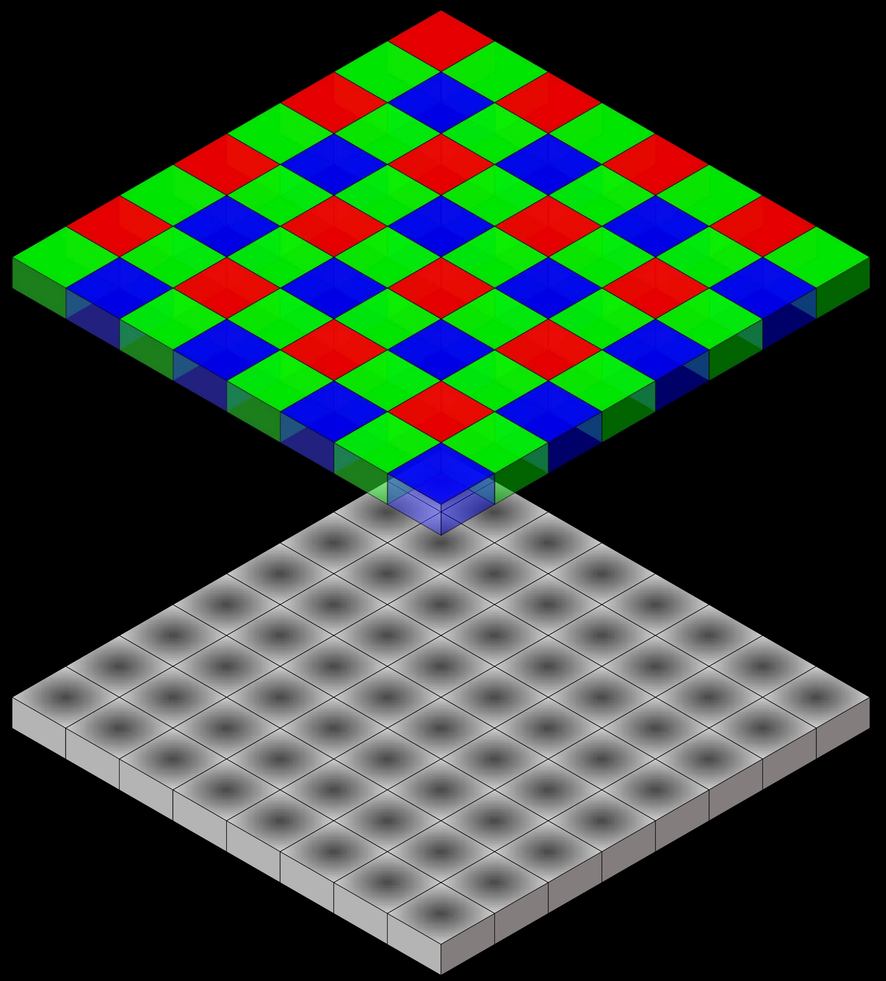
What is RAW?



What is RAW?

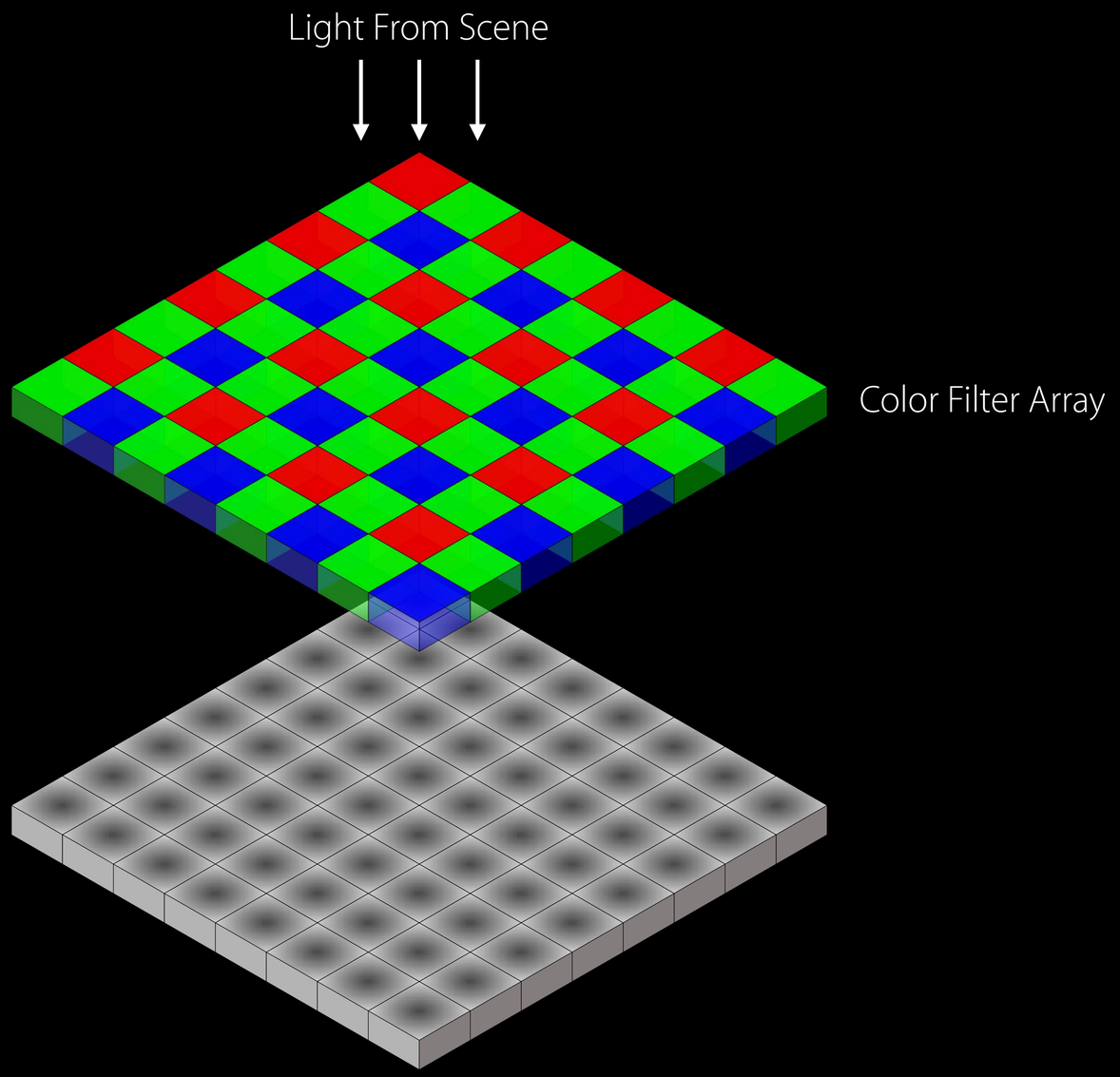


What is RAW?

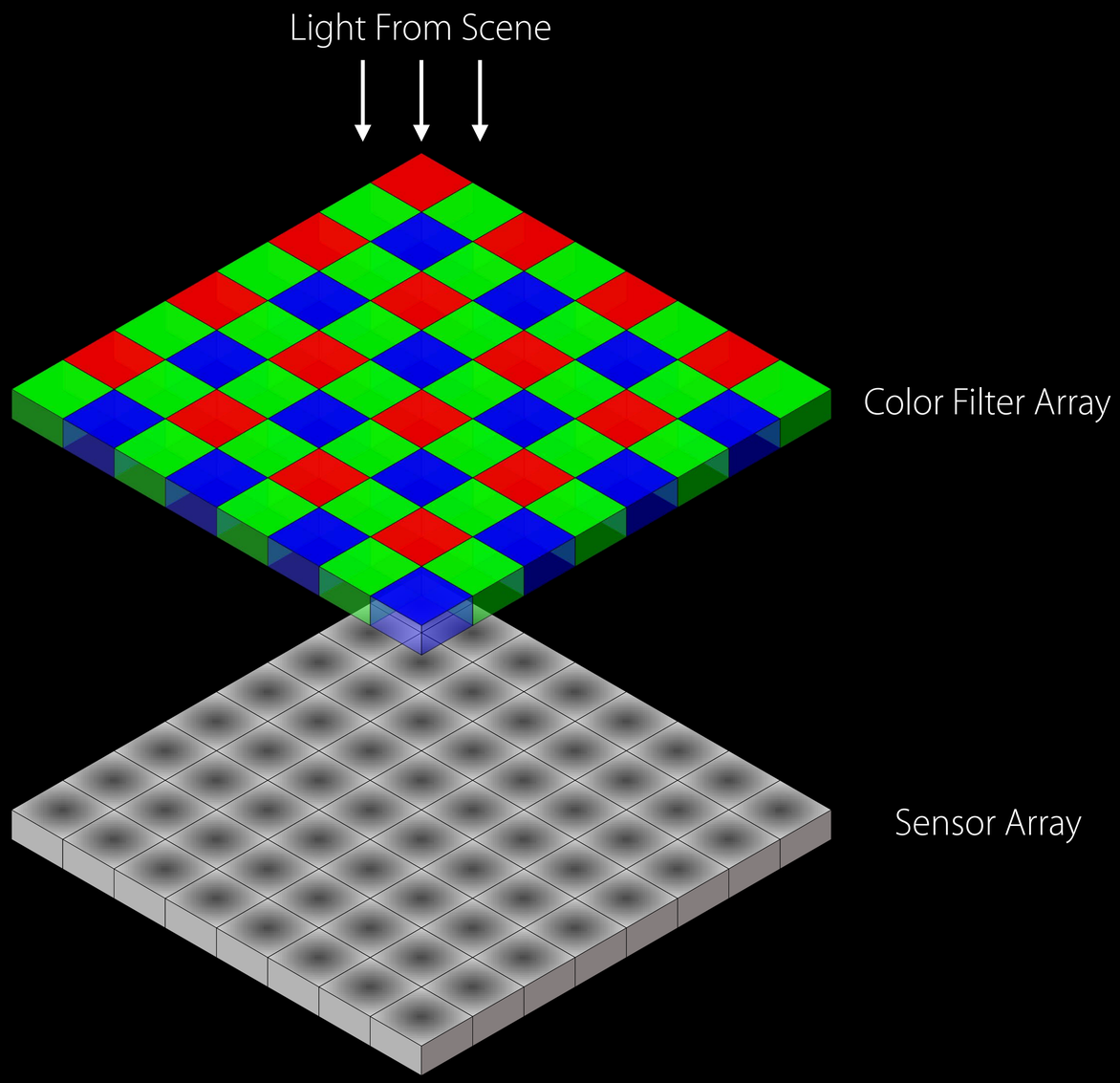


Color Filter Array

What is RAW?



What is RAW?



What is Stored in a RAW file?

What is Stored in a RAW file?

Intensity of red, green, or blue light hitting the sensor

What is Stored in a RAW file?

Intensity of red, green, or blue light hitting the sensor

Bayer pattern as metadata

What is Stored in a RAW file?

Intensity of red, green, or blue light hitting the sensor

Bayer pattern as metadata

A lot of other metadata

What Do RAW Converters Do?

What Do RAW Converters Do?

Demosaic Bayer filter pattern

Apply white balance

Interpret Colorimetric Info

Correct Gamma

Reduce Noise

Sharpen

Why RAW?

Why RAW?

Bake-time flexibility

Why RAW?

Bake-time flexibility

No compression

Why RAW?

Bake-time flexibility

No compression

More bits

Why RAW?

Bake-time flexibility

No compression

More bits

Lots of headroom for editing

Why RAW?

Bake-time flexibility

No compression

More bits

Lots of headroom for editing

Greater artistic freedom to interpret the image data in post

Why JPEG?

Why JPEG?

Lovingly baked by Apple

Why JPEG?

Lovingly baked by Apple

Much faster rendering

Why JPEG?

Lovingly baked by Apple

Much faster rendering

Multiple image fusion (e.g. for stabilization)

Why JPEG?

Lovingly baked by Apple

Much faster rendering

Multiple image fusion (e.g. for stabilization)

Smaller file size

```
// CVPixelBuffer.h

   /* Bayer 14-bit Little-Endian, packed in 16-bits, ordered G R G R...
      alternating with B G B G... */
   kCVPixelFormatType_14Bayer_GRBG         = OSType(“grb4”),

   /* Bayer 14-bit Little-Endian, packed in 16-bits, ordered R G R G...
      alternating with G B G B... */
   kCVPixelFormatType_14Bayer_RGGB         = OSType(“rgg4”),

   /* Bayer 14-bit Little-Endian, packed in 16-bits, ordered B G B G...
      alternating with G R G R... */
   kCVPixelFormatType_14Bayer_BGGR         = OSType(“bgg4”),

   /* Bayer 14-bit Little-Endian, packed in 16-bits, ordered G B G B...
      alternating with R G R G... */
   kCVPixelFormatType_14Bayer_GBRG         = OSType(“gbr4”),
```

Capturing RAW with AVCapturePhotoOutput

Capturing RAW with AVCapturePhotoOutput

RAW is only supported when using a photo format (AVCaptureSessionPresetPhoto)

Capturing RAW with AVCapturePhotoOutput

RAW is only supported when using a photo format (AVCaptureSessionPresetPhoto)

Rear camera only

Capturing RAW with AVCapturePhotoOutput

RAW is only supported when using a photo format (AVCaptureSessionPresetPhoto)

Rear camera only

RAW bracketed captures are supported

```
// Capturing RAW Photos

func takeRawPhoto()
{
    let rawFormat = photoOutput.availableRawPhotoPixelFormatTypes.first!.uint32Value
    let settings = AVCapturePhotoSettings(rawPixelFormatType: rawFormat)

    // RAW photo settings have autoStillImageStabilizationEnabled set to NO
    // highResolutionPhotoEnabled is also NO as it’s meaningless in RAW

    photoOutput.capturePhoto(with: settings, delegate: self)
}
```

```
// Capturing RAW Photos

func takeRawPhoto()
{
    let rawFormat = photoOutput.availableRawPhotoPixelFormatTypes.first!.uint32Value
    let settings = AVCapturePhotoSettings(rawPixelFormatType: rawFormat)

    // RAW photo settings have autoStillImageStabilizationEnabled set to NO
    // highResolutionPhotoEnabled is also NO as it’s meaningless in RAW

    photoOutput.capturePhoto(with: settings, delegate: self)
}
```

```
// Capturing RAW Photos

func takeRawPhoto()
{
    let rawFormat = photoOutput.availableRawPhotoPixelFormatTypes.first!.uint32Value
    let settings = AVCapturePhotoSettings(rawPixelFormatType: rawFormat)

    // RAW photo settings have autoStillImageStabilizationEnabled set to NO
    // highResolutionPhotoEnabled is also NO as it’s meaningless in RAW

    photoOutput.capturePhoto(with: settings, delegate: self)
}
```

```
// Capturing RAW Photos

func takeRawPhoto()
{
    let rawFormat = photoOutput.availableRawPhotoPixelFormatTypes.first!.uint32Value
    let settings = AVCapturePhotoSettings(rawPixelFormatType: rawFormat)

    // RAW photo settings have autoStillImageStabilizationEnabled set to NO
    // highResolutionPhotoEnabled is also NO as it’s meaningless in RAW

    photoOutput.capturePhoto(with: settings, delegate: self)
}
```

```swift
// RAW related AVCapturePhotoCaptureDelegate methods

func capture(_ captureOutput: AVCapturePhotoOutput,
             didFinishProcessingRawPhotoSampleBuffer rawSampleBuffer: CMSampleBuffer?,
             previewPhotoSampleBuffer: CMSampleBuffer?,
             resolvedSettings: AVCaptureResolvedPhotoSettings,
             bracketSettings: AVCaptureBracketedStillImageSettings?,
             error: NSError?)
{
    // Handle the RAW sample buffer
}
```

```
// RAW related AVCapturePhotoCaptureDelegate methods

func capture(_ captureOutput: AVCapturePhotoOutput,
             didFinishProcessingRawPhotoSampleBuffer rawSampleBuffer: CMSampleBuffer?,
             previewPhotoSampleBuffer: CMSampleBuffer?,
             resolvedSettings: AVCaptureResolvedPhotoSettings,
             bracketSettings: AVCaptureBracketedStillImageSettings?,
             error: NSError?)
{
    // Handle the RAW sample buffer
}
```

```swift
// RAW related AVCapturePhotoCaptureDelegate methods

func capture(_ captureOutput: AVCapturePhotoOutput,
             didFinishProcessingRawPhotoSampleBuffer rawSampleBuffer: CMSampleBuffer?,
             previewPhotoSampleBuffer: CMSampleBuffer?,
             resolvedSettings: AVCaptureResolvedPhotoSettings,
             bracketSettings: AVCaptureBracketedStillImageSettings?,
             error: NSError?)
{
    // Handle the RAW sample buffer
}
```

RAW + Processed Image Support

RAW + Processed Image Support

Processed image may be JPEG, or an uncompressed format

RAW + Processed Image Support

Processed image may be JPEG, or an uncompressed format

The processed image is delivered to didFinishProcessingPhotoSampleBuffer:

RAW + Processed Image Support

Processed image may be JPEG, or an uncompressed format

The processed image is delivered to didFinishProcessingPhotoSampleBuffer:

The RAW image is delivered to didFinishProcessingRAWPhotoSampleBuffer:

RAW + Processed Image Support

Processed image may be JPEG, or an uncompressed format

The processed image is delivered to didFinishProcessingPhotoSampleBuffer:

The RAW image is delivered to didFinishProcessingRAWPhotoSampleBuffer:

RAW + processed brackets are supported

RAW + Processed Image Support

Processed image may be JPEG, or an uncompressed format

The processed image is delivered to didFinishProcessingPhotoSampleBuffer:

The RAW image is delivered to didFinishProcessingRAWPhotoSampleBuffer:

RAW + processed brackets are supported

RAW + still image stabilization processed image is NOT supported

```
// Capturing RAW Photos

func takeRAWPlusJPEGPhoto()
{
    let rawFormat = photoOutput.availableRawPhotoPixelFormatTypes.first!.uint32Value
    let processedFormat = photoOutput.availablePhotoCodecTypes.first!
    let settings = AVCapturePhotoSettings(rawPixelFormatType: rawFormat, processedFormat:
                                          [AVVideoCodecKey : processedFormat])

    // highResolutionPhotoEnabled YES or NO applies only to the processed photo

    photoOutput.capturePhoto(with: settings, delegate: self)
}
```

```
// Capturing RAW Photos

func takeRAWPlusJPEGPhoto()
{
    let rawFormat = photoOutput.availableRawPhotoPixelFormatTypes.first!.uint32Value
    let processedFormat = photoOutput.availablePhotoCodecTypes.first!
    let settings = AVCapturePhotoSettings(rawPixelFormatType: rawFormat, processedFormat:
                                          [AVVideoCodecKey : processedFormat])

    // highResolutionPhotoEnabled YES or NO applies only to the processed photo

    photoOutput.capturePhoto(with: settings, delegate: self)
}
```

```
// Capturing RAW Photos

func takeRAWPlusJPEGPhoto()
{
    let rawFormat = photoOutput.availableRawPhotoPixelFormatTypes.first!.uint32Value
    let processedFormat = photoOutput.availablePhotoCodecTypes.first!
    let settings = AVCapturePhotoSettings(rawPixelFormatType: rawFormat, processedFormat:
                                          [AVVideoCodecKey : processedFormat])

    // highResolutionPhotoEnabled YES or NO applies only to the processed photo

    photoOutput.capturePhoto(with: settings, delegate: self)
}
```

Storing RAW Buffers

```
// Writing RAW to DNG

func capture(_ captureOutput: AVCapturePhotoOutput,
                didFinishProcessingRawPhotoSampleBuffer rawSampleBuffer: CMSampleBuffer?,
                previewPhotoSampleBuffer: CMSampleBuffer?,
                resolvedSettings: AVCaptureResolvedPhotoSettings,
                bracketSettings: AVCaptureBracketedStillImageSettings?,
                error: NSError?)
{
    if let rawSampleBuffer = rawSampleBuffer,
        data = AVCapturePhotoOutput.dngPhotoDataRepresentation(forRawSampleBuffer:
                        rawSampleBuffer, previewPhotoSampleBuffer: previewPhotoSampleBuffer)
    {
        let filePath = (_uniqueFilePath as NSString).appendingPathExtension(".dng")!
        do {
            try data.write(to: URL(fileURLWithPath: filePath), options: .atomicWrite)
        }
        catch {
            // Handle error
        }
    }
}
```

```
// Writing RAW to DNG

func capture(_ captureOutput: AVCapturePhotoOutput,
                didFinishProcessingRawPhotoSampleBuffer rawSampleBuffer: CMSampleBuffer?,
                previewPhotoSampleBuffer: CMSampleBuffer?,
                resolvedSettings: AVCaptureResolvedPhotoSettings,
                bracketSettings: AVCaptureBracketedStillImageSettings?,
                error: NSError?)
{
    if let rawSampleBuffer = rawSampleBuffer,
        data = AVCapturePhotoOutput.dngPhotoDataRepresentation(forRawSampleBuffer:
                    rawSampleBuffer, previewPhotoSampleBuffer: previewPhotoSampleBuffer)
    {
        let filePath = (_uniqueFilePath as NSString).appendingPathExtension(".dng")!
        do {
            try data.write(to: URL(fileURLWithPath: filePath), options: .atomicWrite)
        }
        catch {
            // Handle error
        }
    }
}
```

Demo

RAW Capture and Editing

AVCamManual and RawExpose

More on RAW Processing in Session 505

Live Photo Editing and RAW Processing with Core Image	Nob Hill	Thursday 11:00AM

RAW Photo Capture Support

RAW Photo Capture Support

iPhone 6s

iPhone 6s Plus

iPhone SE

9.7-inch iPad Pro

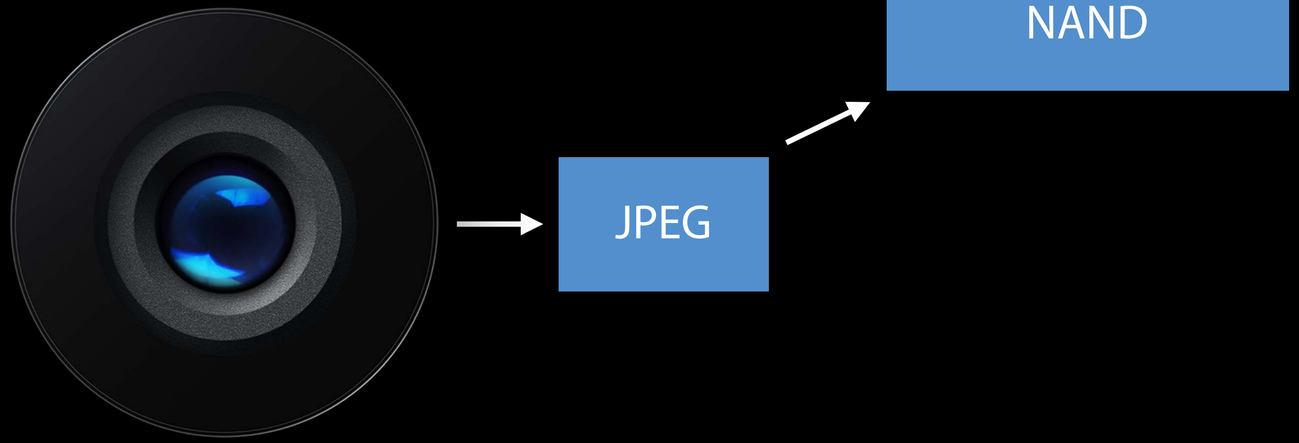
Capturing Preview Images
a.k.a. Thumbnails

HDR

SLO-MO VIDEO PHOTO SQUARE PANO

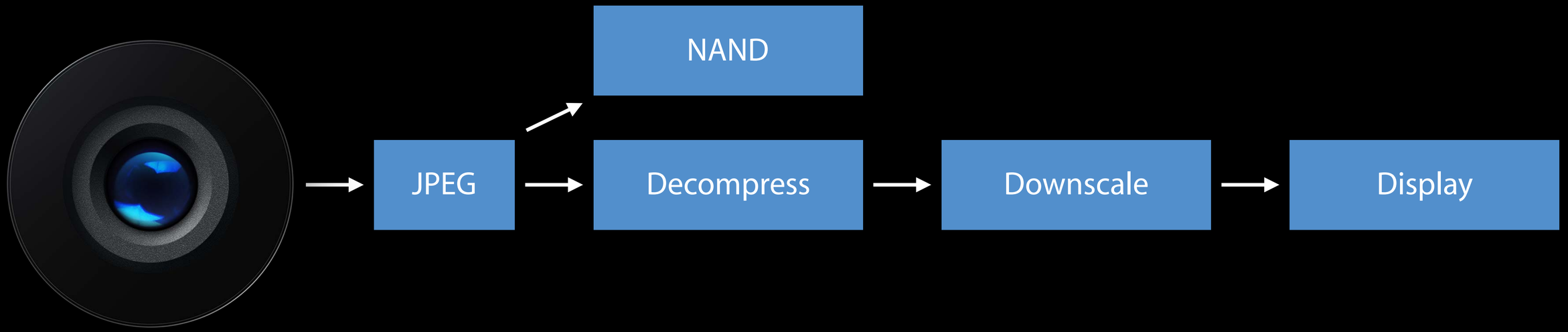
HDR

SLO-MO VIDEO PHOTO SQUARE PANO



JPEG

NAND


NAND
JPEG
Decompress
Downscale
Display

NAND

JPEG

Decompress

Downscale

Display

Decompressed, downscaled image

JPEG

NAND

Decompressed, downscaled image

Display

Preview Images

Preview Images

didFinishProcessing{Raw}PhotoSampleBuffer: callback can deliver a preview image

Preview Images

didFinishProcessing{Raw}PhotoSampleBuffer: callback can deliver a preview image

Preview image is uncompressed

Preview Images

didFinishProcessing{Raw}PhotoSampleBuffer: callback can deliver a preview image

Preview image is uncompressed

You can specify the preview image dimensions

Preview Images

didFinishProcessing{Raw}PhotoSampleBuffer: callback can deliver a preview image

Preview image is uncompressed

You can specify the preview image dimensions

Photo output can select a size for you

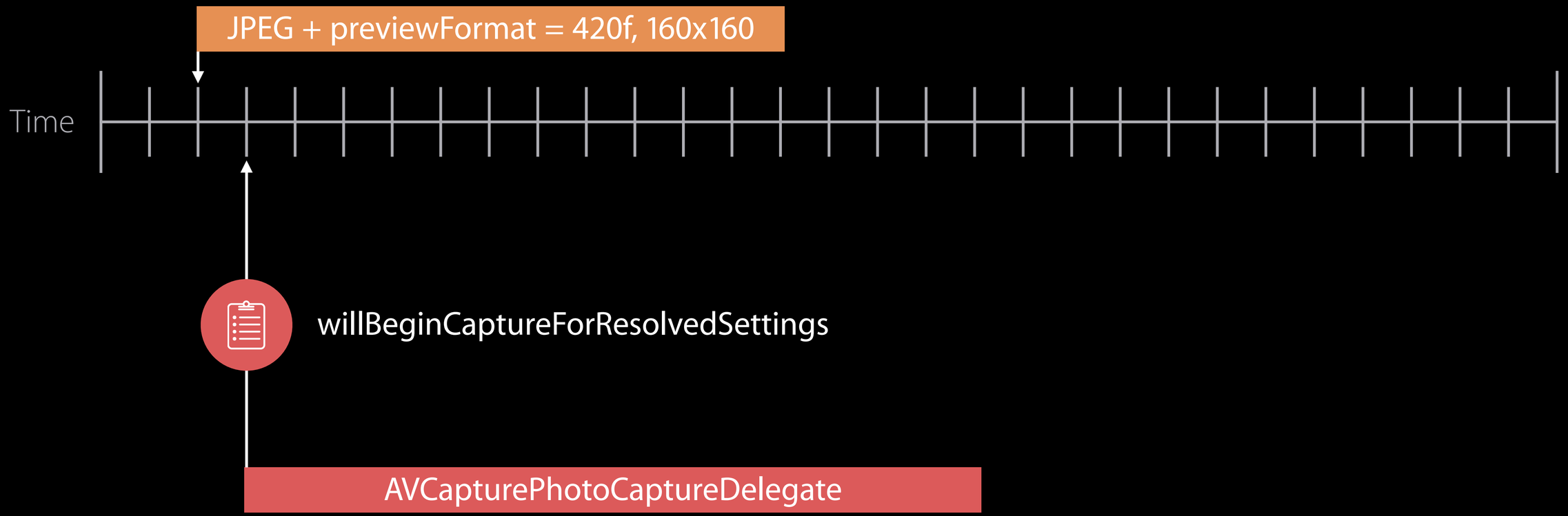
```swift
// Capturing a processed image with a preview (thumbnail)

func takePhotoWithPreviewImage()
{
   let settings = AVCapturePhotoSettings()
   settings.isHighResolutionPhotoEnabled = true

   let previewPixelType = settings.availablePreviewPhotoPixelFormatTypes.first!.uint32Value
   let previewFormat = [kCVPixelBufferPixelFormatTypeKey as String :
                        previewPixelType as! AnyObject,
                        kCVPixelBufferWidthKey as String : 160,
                        kCVPixelBufferHeightKey as String : 160]
   settings.previewPhotoFormat = previewFormat

   photoOutput.capturePhoto(with: settings, delegate: self)
}
```

```swift
// Capturing a processed image with a preview (thumbnail)

func takePhotoWithPreviewImage()
{
    let settings = AVCapturePhotoSettings()
    settings.isHighResolutionPhotoEnabled = true

    let previewPixelType = settings.availablePreviewPhotoPixelFormatTypes.first!.uint32Value
    let previewFormat = [kCVPixelBufferPixelFormatTypeKey as String :
                                 previewPixelType as! AnyObject,
                                 kCVPixelBufferWidthKey as String : 160,
                                 kCVPixelBufferHeightKey as String : 160]
    settings.previewPhotoFormat = previewFormat

    photoOutput.capturePhoto(with: settings, delegate: self)
}
```

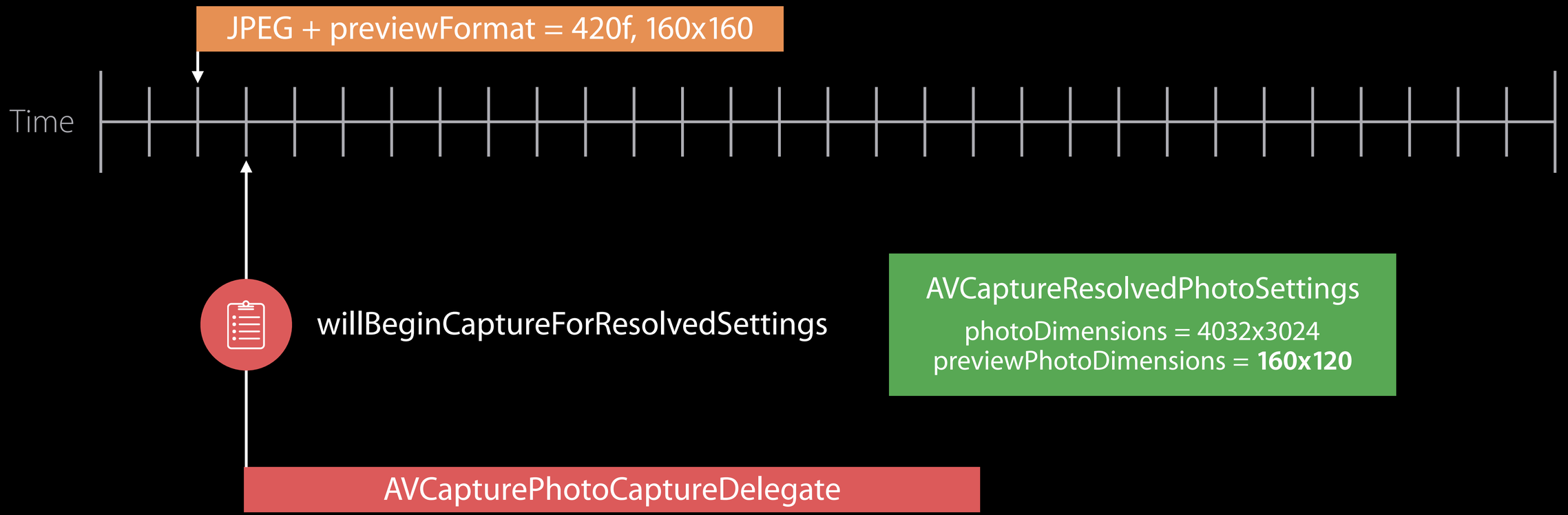
```swift
// Capturing a processed image with a preview (thumbnail)

func takePhotoWithPreviewImage()
{
    let settings = AVCapturePhotoSettings()
    settings.isHighResolutionPhotoEnabled = true

    let previewPixelType = settings.availablePreviewPhotoPixelFormatTypes.first!.uint32Value
    let previewFormat = [kCVPixelBufferPixelFormatTypeKey as String :
                             previewPixelType as! AnyObject,
                         kCVPixelBufferWidthKey as String : 160,
                         kCVPixelBufferHeightKey as String : 160]
    settings.previewPhotoFormat = previewFormat

    photoOutput.capturePhoto(with: settings, delegate: self)
}
```

```
// Capturing a processed image with a preview (thumbnail)

func takePhotoWithPreviewImage()
{
   let settings = AVCapturePhotoSettings()
   settings.isHighResolutionPhotoEnabled = true

   let previewPixelType = settings.availablePreviewPhotoPixelFormatTypes.first!.uint32Value
   let previewFormat = [kCVPixelBufferPixelFormatTypeKey as String :
                        previewPixelType as! AnyObject,
                        kCVPixelBufferWidthKey as String : 160,
                        kCVPixelBufferHeightKey as String : 160]
   settings.previewPhotoFormat = previewFormat

   photoOutput.capturePhoto(with: settings, delegate: self)
}
```

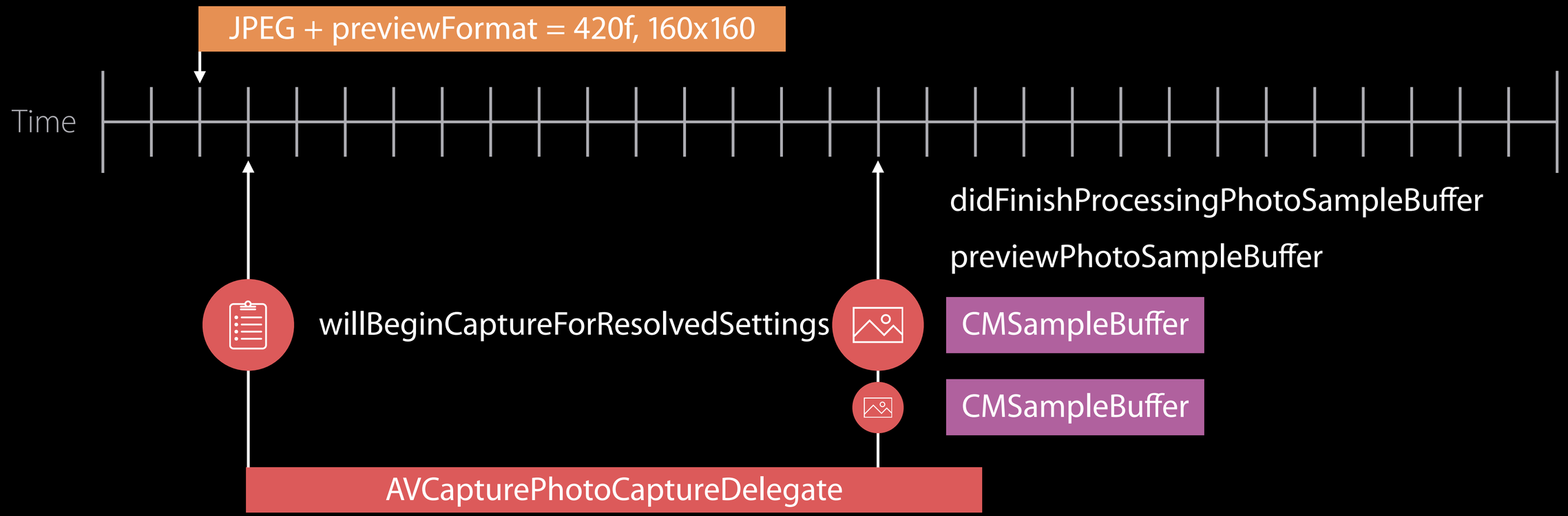
Preview Image Retrieval



Preview Image Retrieval



Preview Image Retrieval



```
// Use preview image as a thumbnail

func capture(_ captureOutput: AVCapturePhotoOutput,
                didFinishProcessingRawPhotoSampleBuffer rawSampleBuffer: CMSampleBuffer?,
                previewPhotoSampleBuffer: CMSampleBuffer?,
                resolvedSettings: AVCaptureResolvedPhotoSettings,
                bracketSettings: AVCaptureBracketedStillImageSettings?,
                error: NSError?)
{
    if let rawSampleBuffer = rawSampleBuffer,
        data = AVCapturePhotoOutput.dngPhotoDataRepresentation(forRawSampleBuffer:
                        rawSampleBuffer, previewPhotoSampleBuffer: previewPhotoSampleBuffer)
    {
        let filePath = (_uniqueFilePath as NSString).appendingPathExtension(".dng")!
        do {
            try data.write(to: URL(fileURLWithPath: filePath), options: .atomicWrite)
        }
        catch {
            // Handle error
        }
    }
}
```

```
// Use preview image as a thumbnail

func capture(_ captureOutput: AVCapturePhotoOutput,
                 didFinishProcessingRawPhotoSampleBuffer rawSampleBuffer: CMSampleBuffer?,
                 previewPhotoSampleBuffer: CMSampleBuffer?,
                 resolvedSettings: AVCaptureResolvedPhotoSettings,
                 bracketSettings: AVCaptureBracketedStillImageSettings?,
                 error: NSError?)
{
    if let rawSampleBuffer = rawSampleBuffer,
        data = AVCapturePhotoOutput.dngPhotoDataRepresentation(forRawSampleBuffer:
                        rawSampleBuffer, previewPhotoSampleBuffer: previewPhotoSampleBuffer)
    {
        let filePath = (_uniqueFilePath as NSString).appendingPathExtension(".dng")!
        do {
            try data.write(to: URL(fileURLWithPath: filePath), options: .atomicWrite)
        }
        catch {
            // Handle error
        }
    }
}
```

```swift
// Use preview image as a thumbnail

func capture(_ captureOutput: AVCapturePhotoOutput,
                didFinishProcessingRawPhotoSampleBuffer rawSampleBuffer: CMSampleBuffer?,
                previewPhotoSampleBuffer: CMSampleBuffer?,
                resolvedSettings: AVCaptureResolvedPhotoSettings,
                bracketSettings: AVCaptureBracketedStillImageSettings?,
                error: NSError?)
{
    if let rawSampleBuffer = rawSampleBuffer,
        data = AVCapturePhotoOutput.dngPhotoDataRepresentation(forRawSampleBuffer:
                        rawSampleBuffer, previewPhotoSampleBuffer: previewPhotoSampleBuffer)
    {
        let filePath = (_uniqueFilePath as NSString).appendingPathExtension(".dng")!
        do {
            try data.write(to: URL(fileURLWithPath: filePath), options: .atomicWrite)
        }
        catch {
            // Handle error
        }
    }
}
```

Preview Image Support

Everywhere!

Wide Color Capture

Brief Overview

Brief Overview

iPad Pro 9.7 True Tone Display

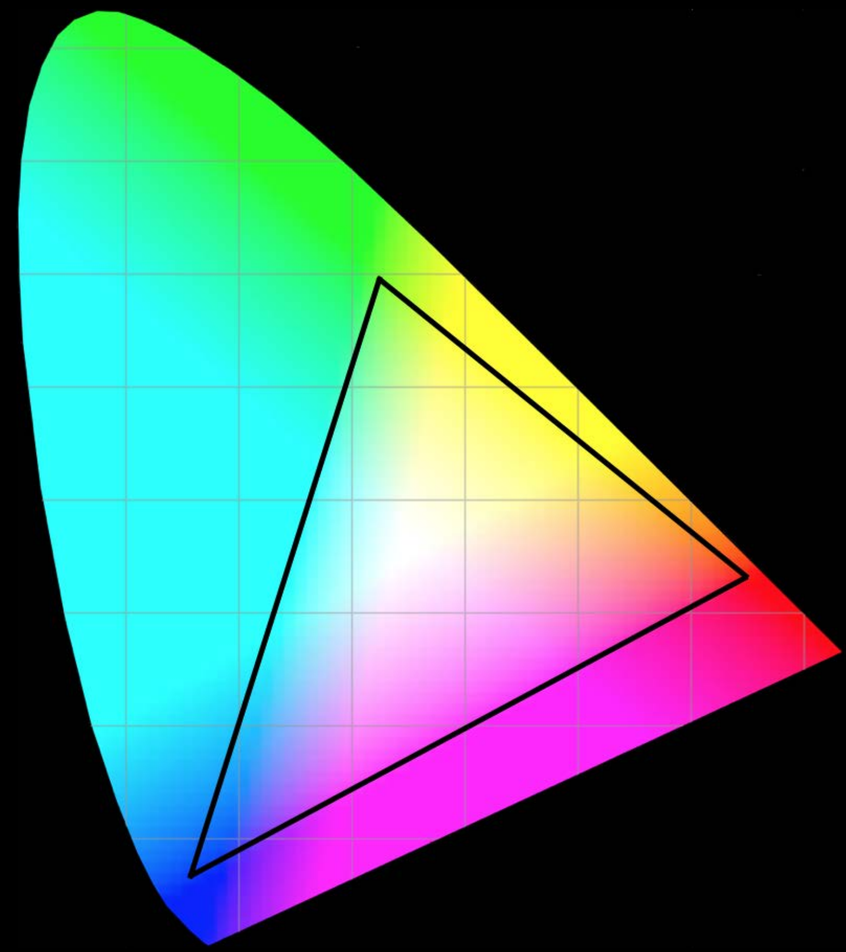
Brief Overview

iPad Pro 9.7 True Tone Display

Color management in iOS 9.3!

Intro to Wide Color

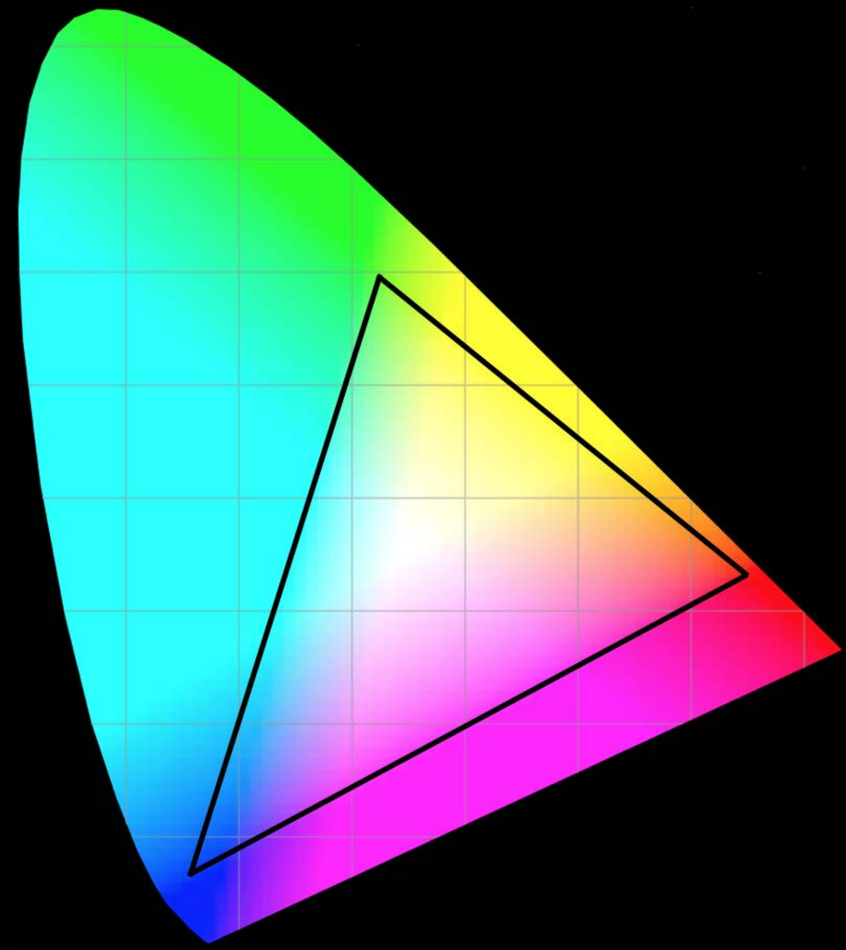
sRGB color space



Intro to Wide Color

sRGB color space

Based on the ITU-R BT.709 standard

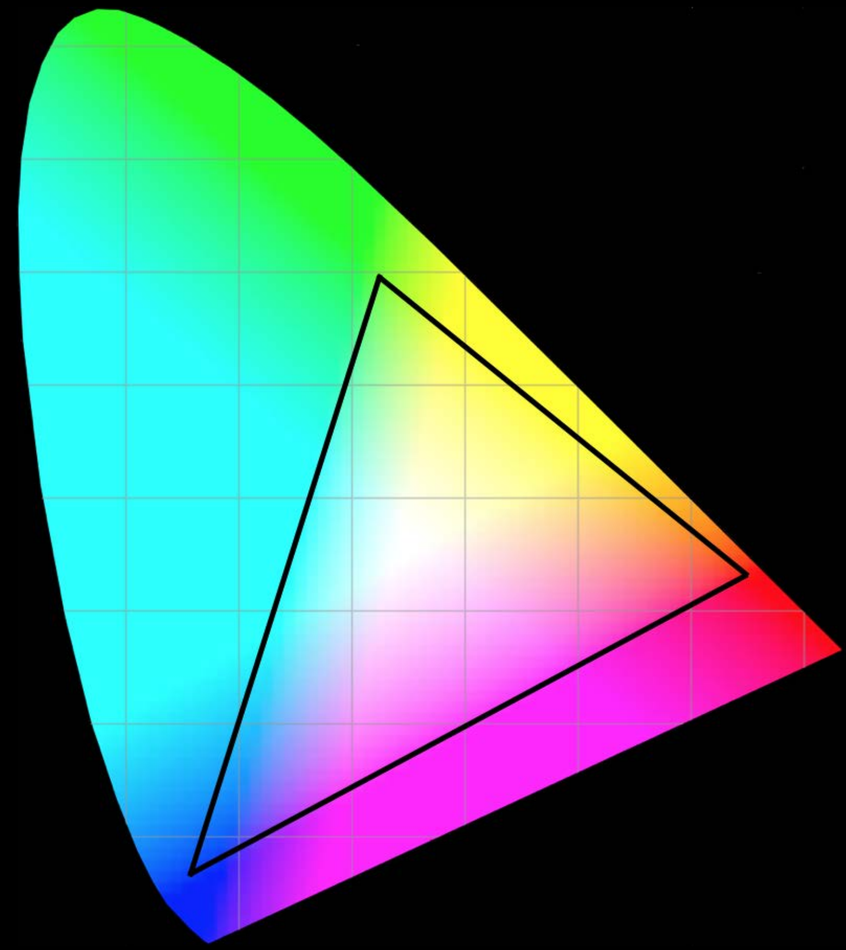


Intro to Wide Color

sRGB color space

Based on the ITU-R BT.709 standard

Gamma ≈ 2.2



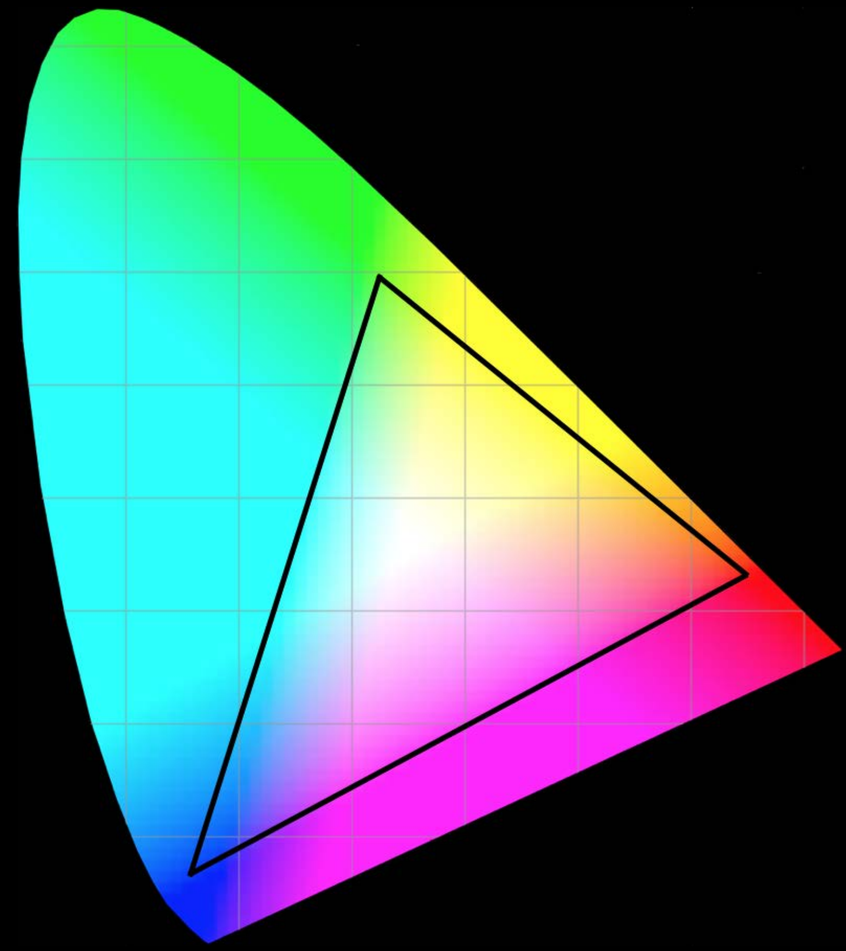
Intro to Wide Color

sRGB color space

Based on the ITU-R BT.709 standard

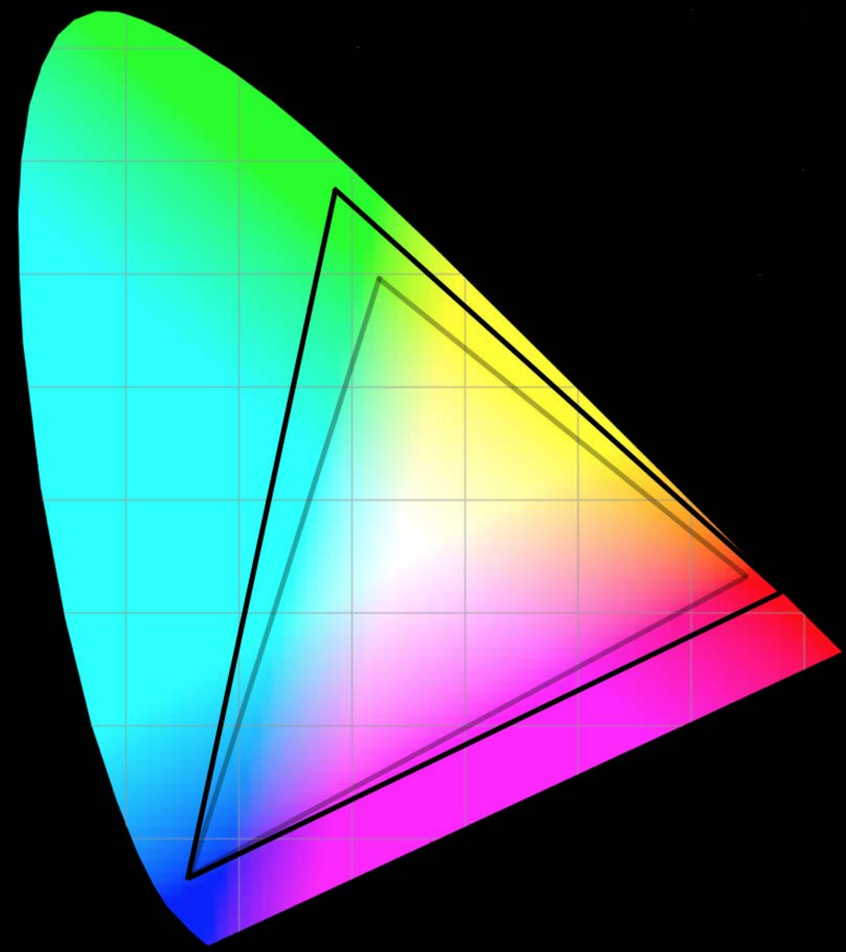
Gamma ≈ 2.2

White point of 6500 degrees K (D65)



Intro to Wide Color

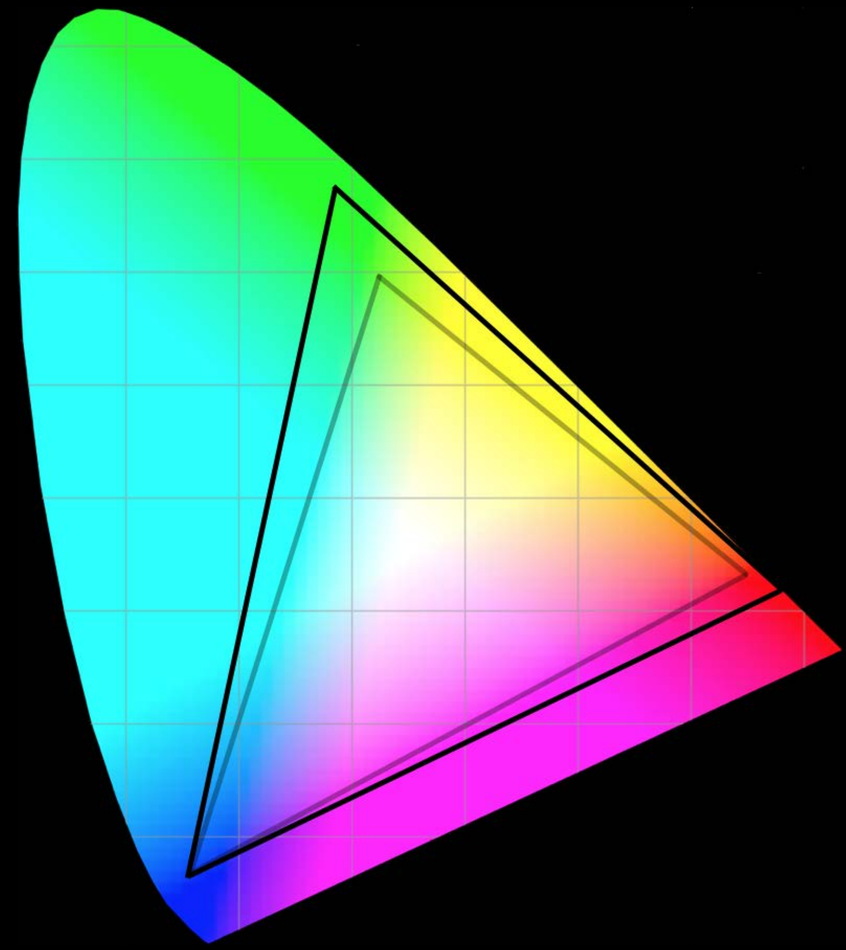
Display P3 color space



Intro to Wide Color

Display P3 color space

Based on the DCI-P3 standard

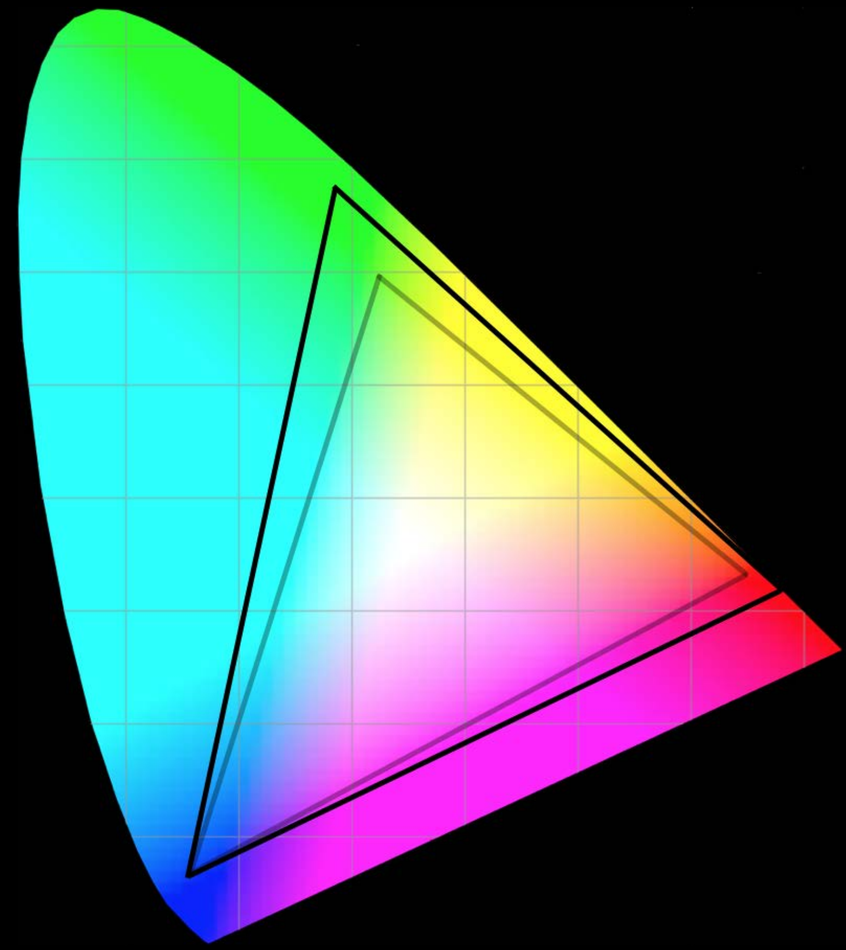


Intro to Wide Color

Display P3 color space

Based on the DCI-P3 standard

Gamma ≈ 2.2



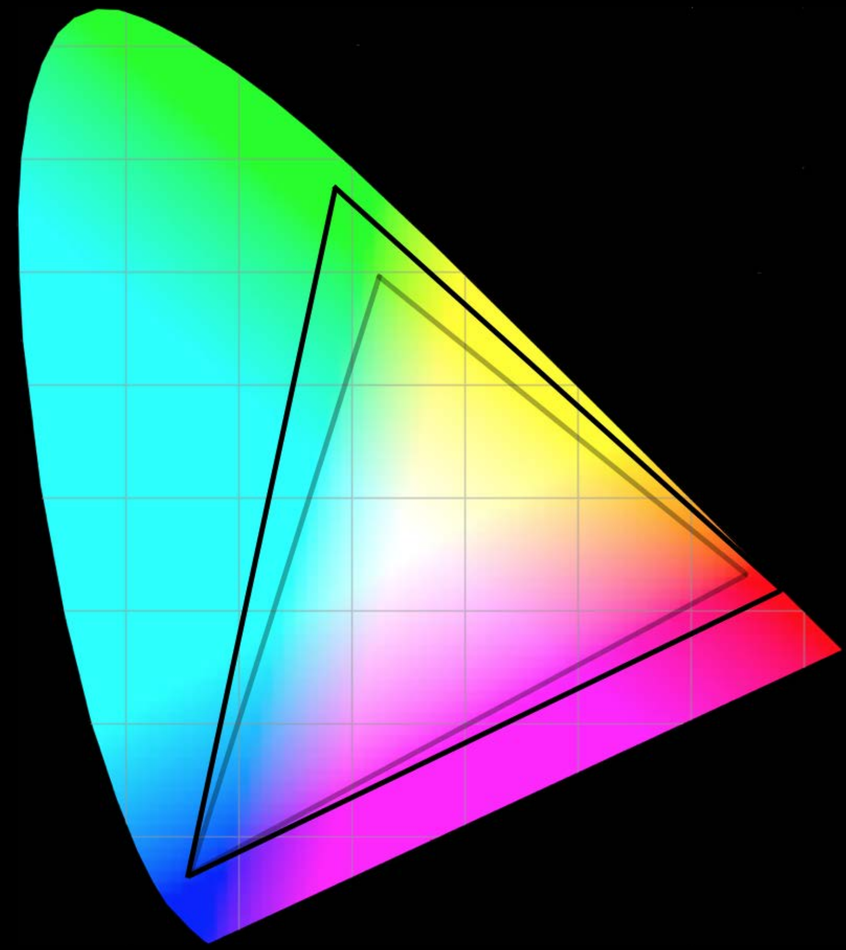
Intro to Wide Color

Display P3 color space

Based on the DCI-P3 standard

Gamma ≈ 2.2

White point of 6500 degrees K (D65)



Profiles

Profile First Aid | Profiles | Devices | Filters | Calculator

Installed ColorSync profiles:

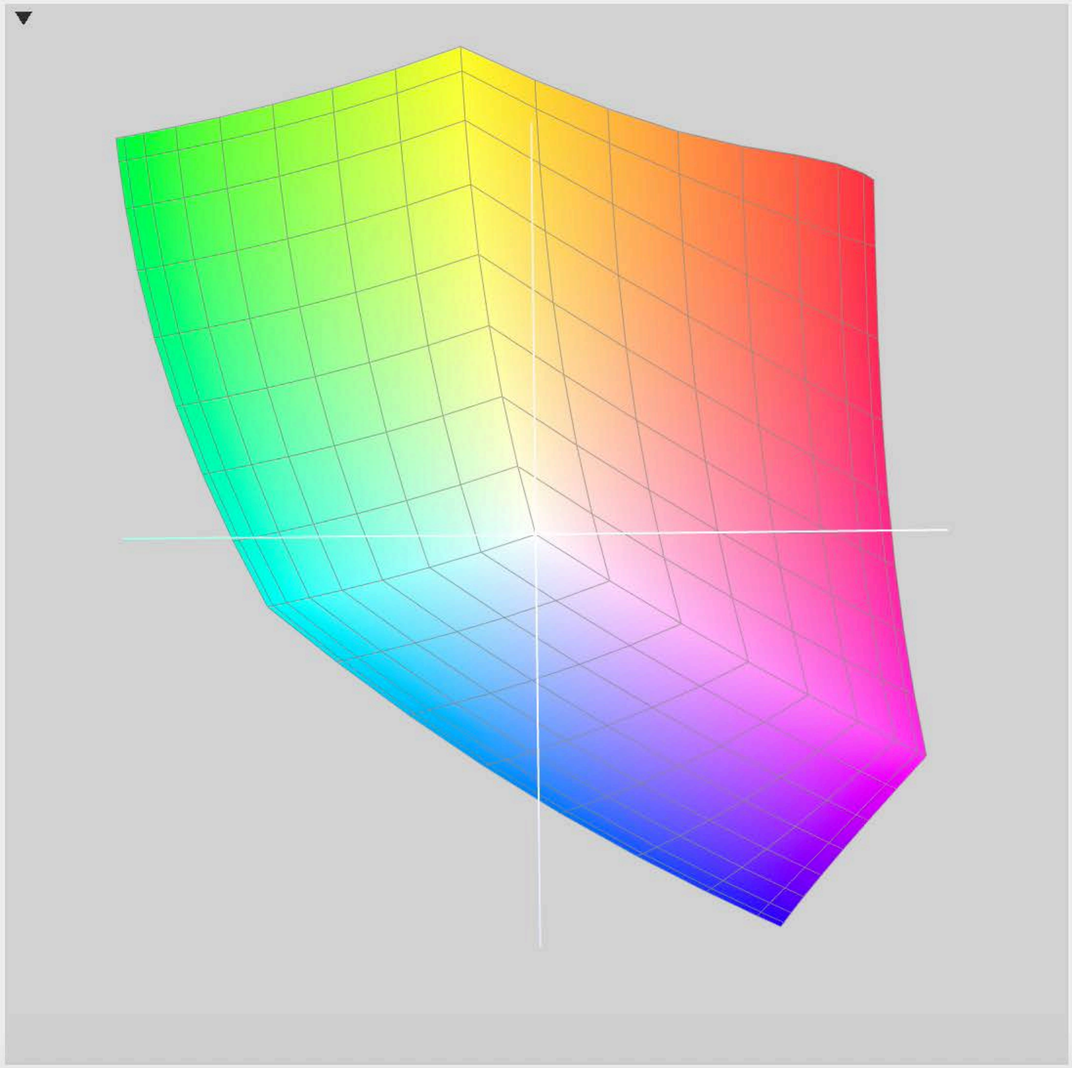
Profile

- System
  - ACES CG Linear (Academy Color
  - Adobe RGB (1998)
  - Display P3
  - Generic CMYK Profile
  - Generic Gray Gamma 2.2 Profile
  - Generic Gray Profile
  - Generic L*a*b* Profile
  - Generic RGB Profile
  - Generic XYZ Profile
  - Rec. ITU-R BT.2020-1
  - Rec. ITU-R BT.709-5
  - ROMM RGB: ISO 22028-2:2013
  - SMPTE RP 431-2-2007 DCI (P3)
  - sRGB IEC61966-2.1
- Computer
  - Black & White
  - Blue Tone
  - Displays
  - Gray Tone
  - Lightness Decrease
  - Lightness Increase
  - Sepia Tone
  - Web Safe Colors
- User
- Other
  - /System/Library/CoreServices/Re
  - /System/Library/Frameworks/ICAI

Profile Information:

Name: **Display P3**
Path: **/System/Library/ColorSync/Profiles/Display P3.icc** Open
Class: **Display**
Space: **RGB**
PCS: **XYZ**
Version: **4.0.0**
Created: **Wednesday, October 14, 2015 at 1:08:57 PM Pacific Daylight Time**
Size: **548 bytes**

Lab Plot:



Profiles

Profile First Aid | Profiles | Devices | Filters | Calculator

Installed ColorSync profiles:

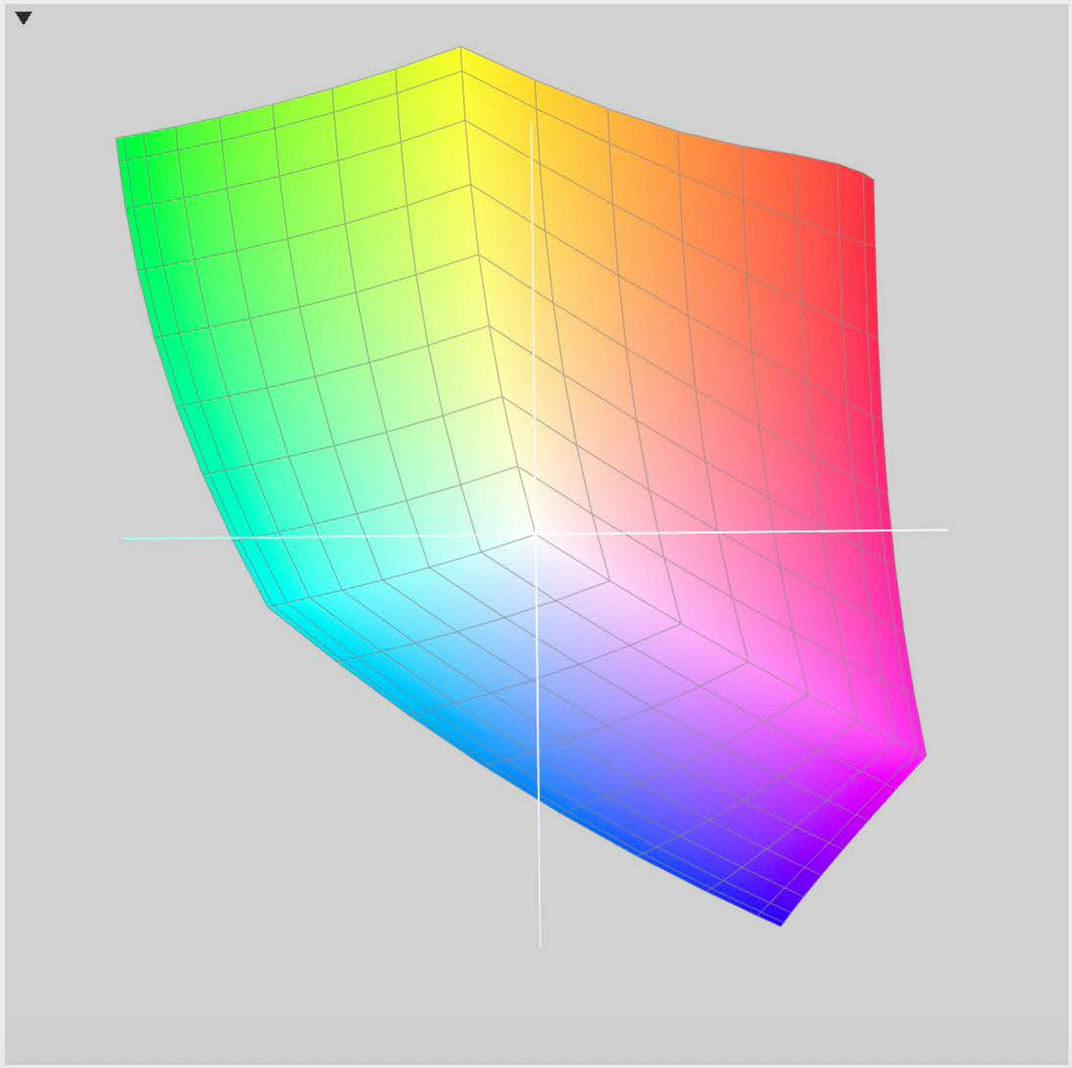
Profile

- System
  - ACES CG Linear (Academy Color
  - Adobe RGB (1998)
  - Display P3
  - Generic CMYK Profile
  - Generic Gray Gamma 2.2 Profile
  - Generic Gray Profile
  - Generic L*a*b* Profile
  - Generic RGB Profile
  - Generic XYZ Profile
  - Rec. ITU-R BT.2020-1
  - Rec. ITU-R BT.709-5
  - ROMM RGB: ISO 22028-2:2013
  - SMPTE RP 431-2-2007 DCI (P3)
  - sRGB IEC61966-2.1
- Computer
  - Black & White
  - Blue Tone
  - Displays
  - Gray Tone
  - Lightness Decrease
  - Lightness Increase
  - Sepia Tone
  - Web Safe Colors
- User
- Other
  - /System/Library/CoreServices/Re
  - /System/Library/Frameworks/ICAI

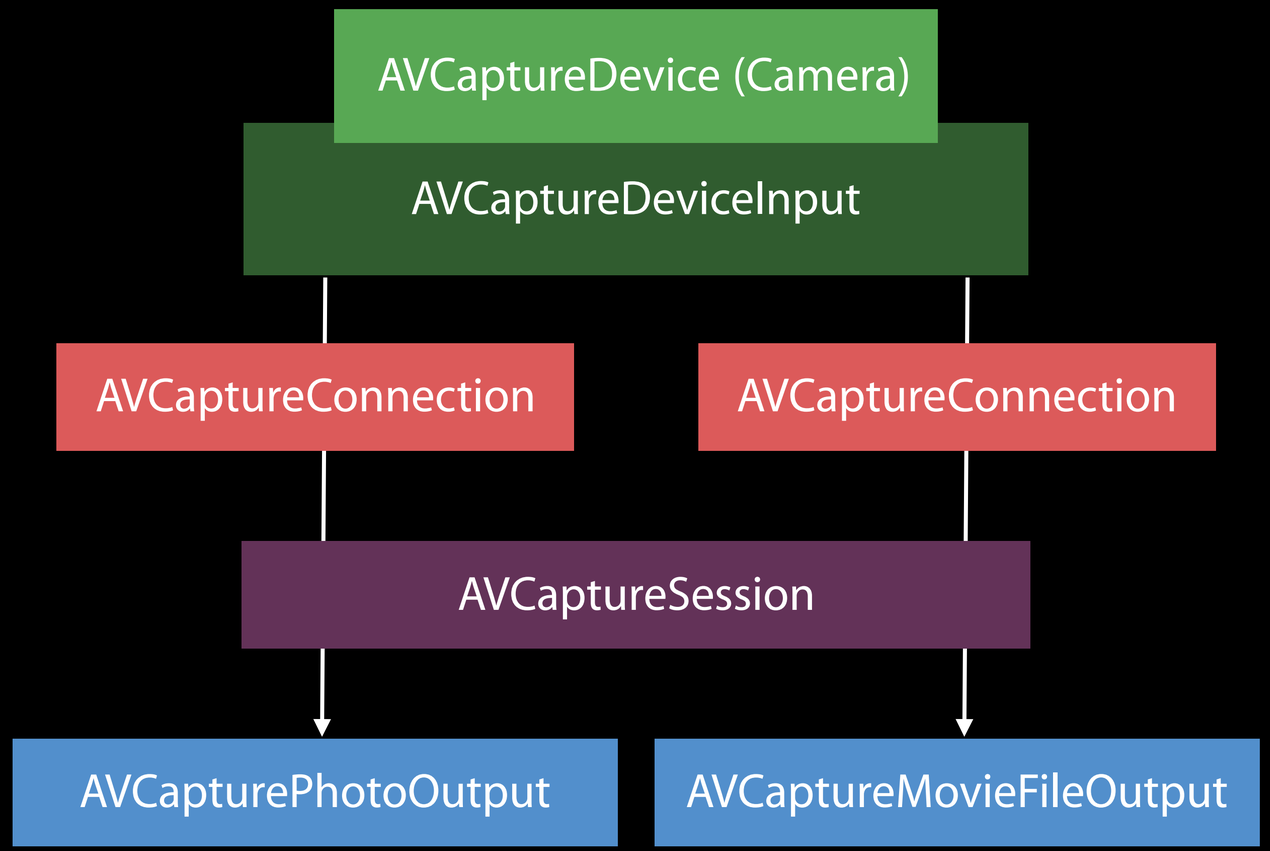
Profile Information:

Name: **Display P3**
Path: **/System/Library/ColorSync/Profiles/Display P3.icc** Open
Class: **Display**
Space: **RGB**
PCS: **XYZ**
Version: **4.0.0**
Created: **Wednesday, October 14, 2015 at 1:08:57 PM Pacific Daylight Time**
Size: **548 bytes**

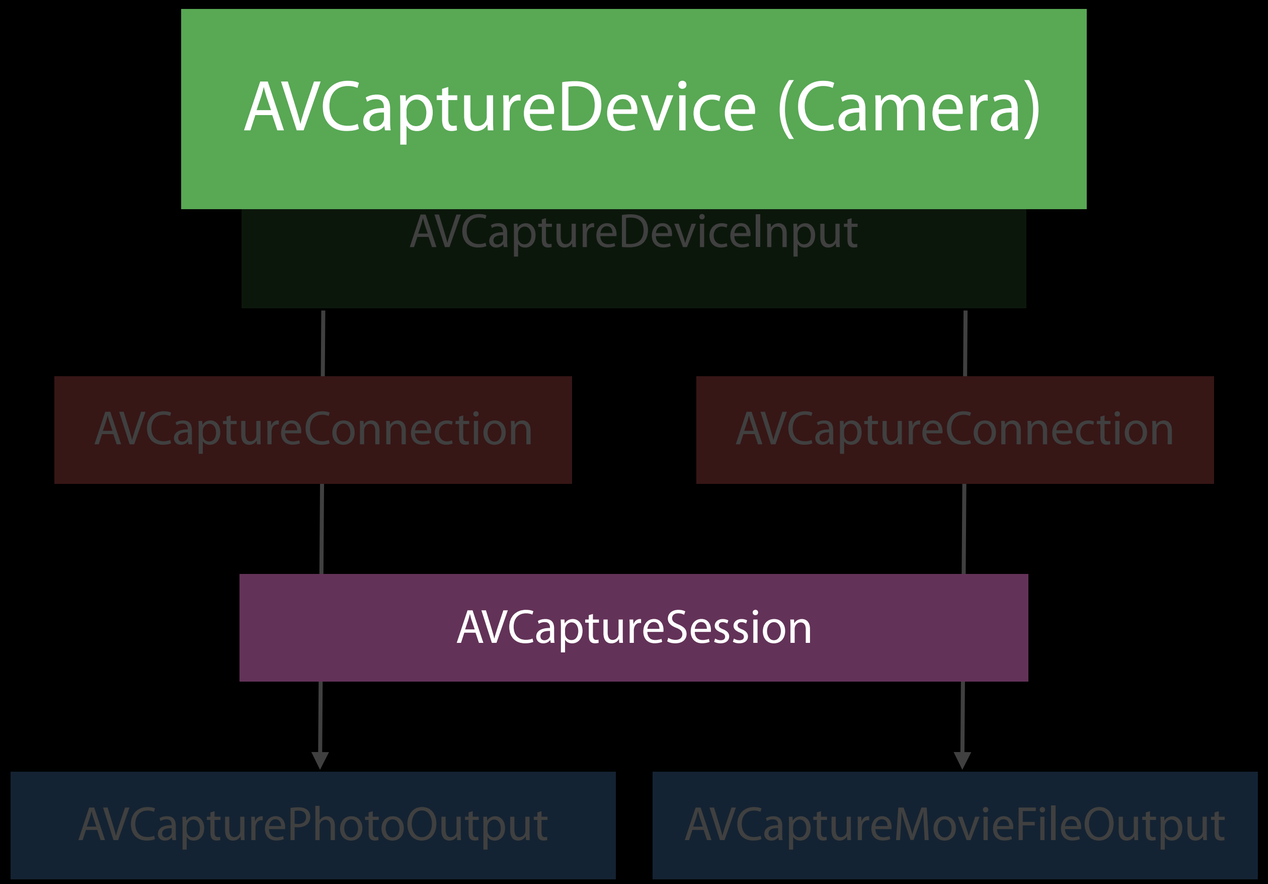
Lab Plot:



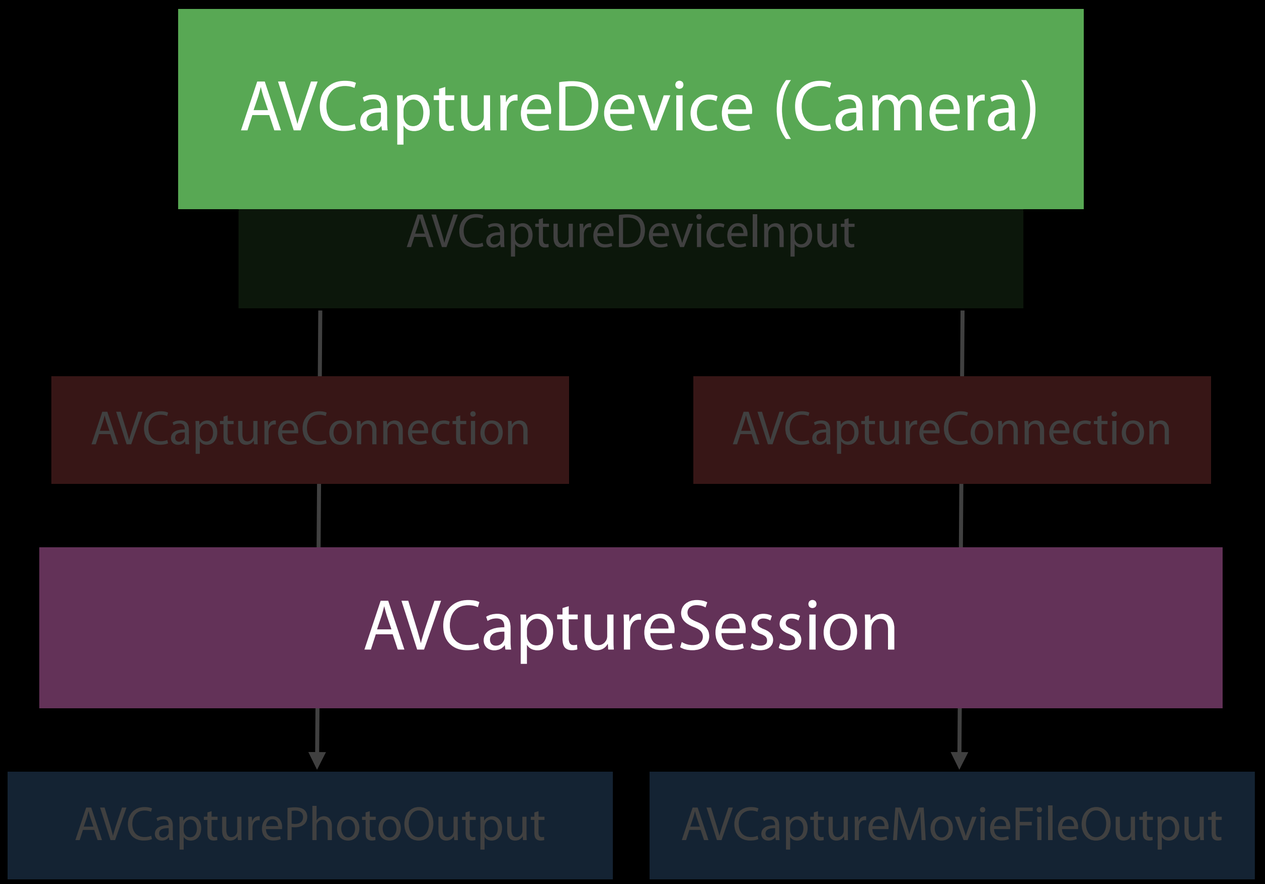
Capturing Display P3 Color on iPad Pro 9.7



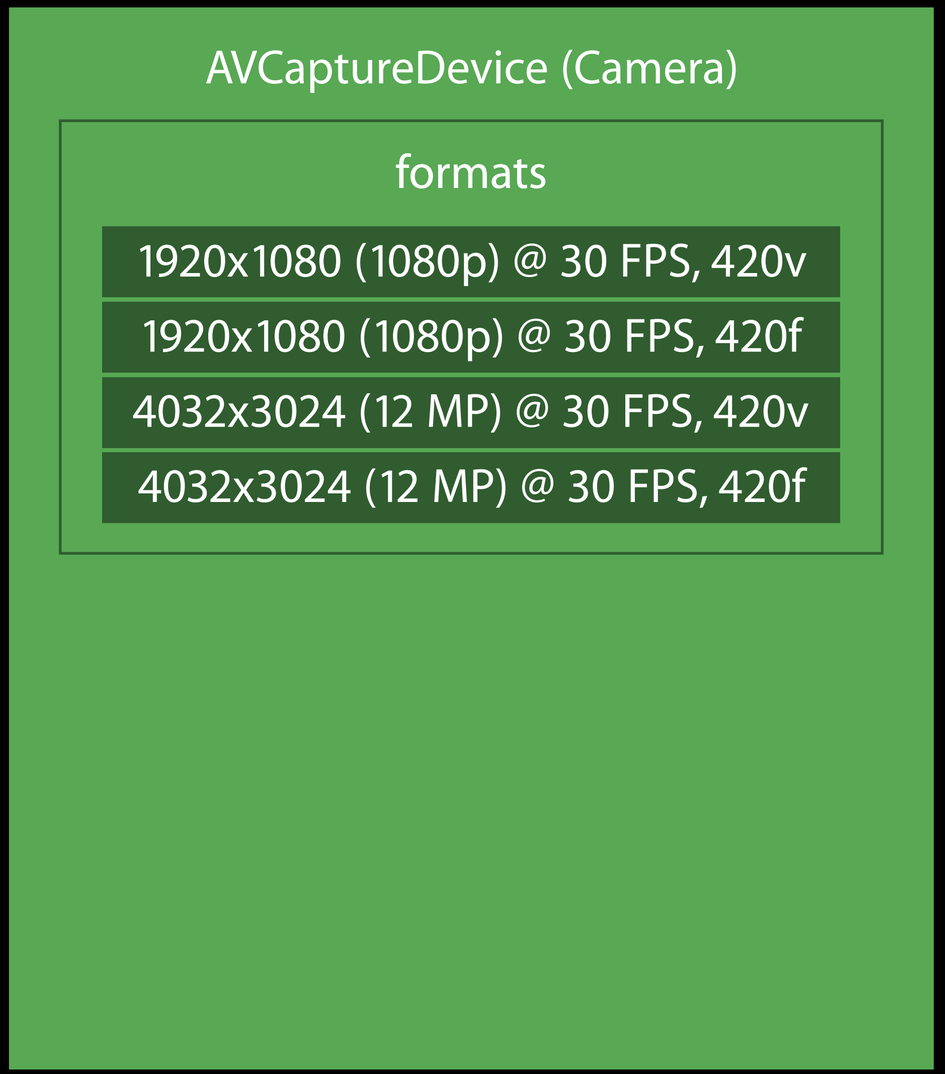
Capturing Display P3 Color on iPad Pro 9.7



Capturing Display P3 Color on iPad Pro 9.7



Capturing Display P3 Color on iPad Pro 9.7



Capturing Display P3 Color on iPad Pro 9.7

AVCaptureDevice (Camera)

formats

1920x1080 (1080p) @ 30 FPS, 420v

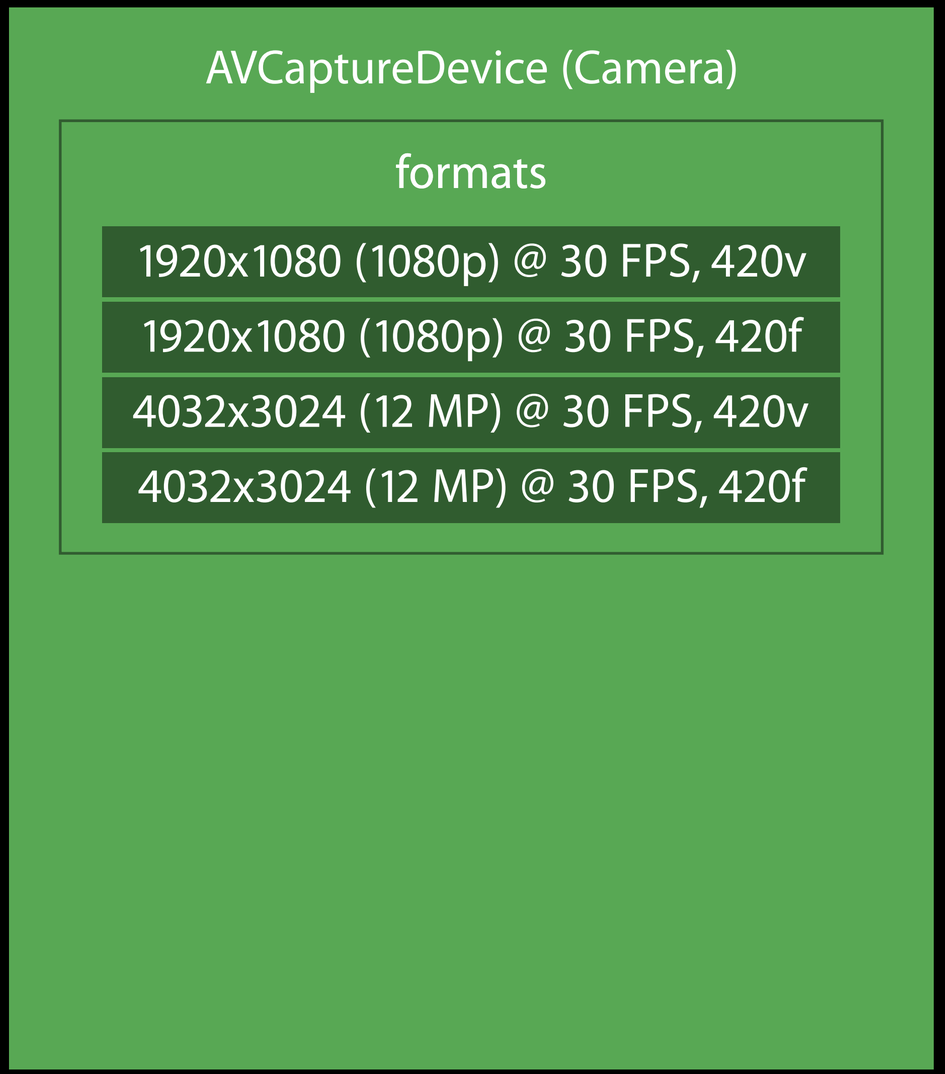
1920x1080 (1

4032x3024 (12

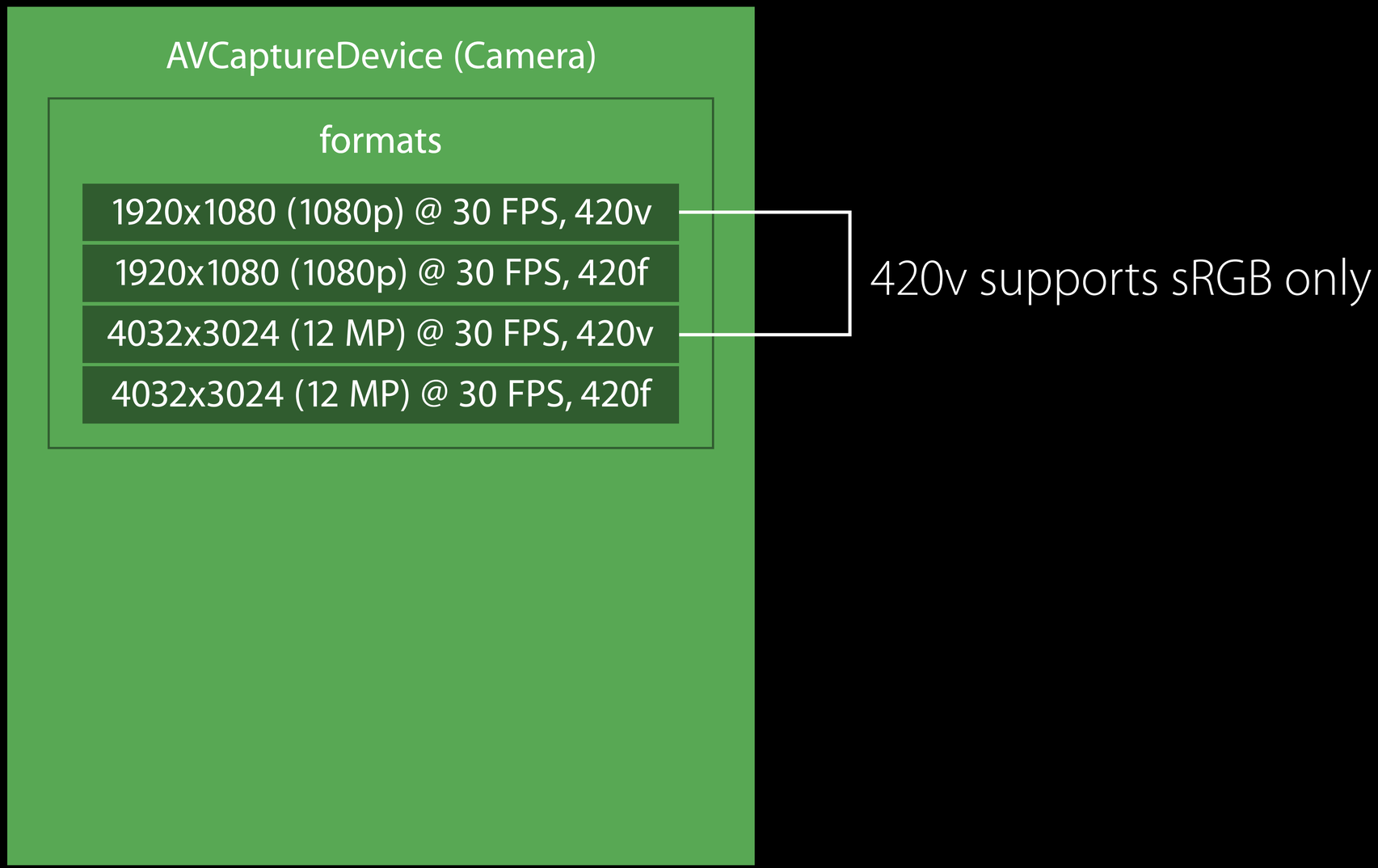
4032x3024 (1

```
public var supportedColorSpaces: [NSNumber]! { get }

public enum AVCaptureColorSpace : Int {
    case sRGB
    case P3_D65
}
```

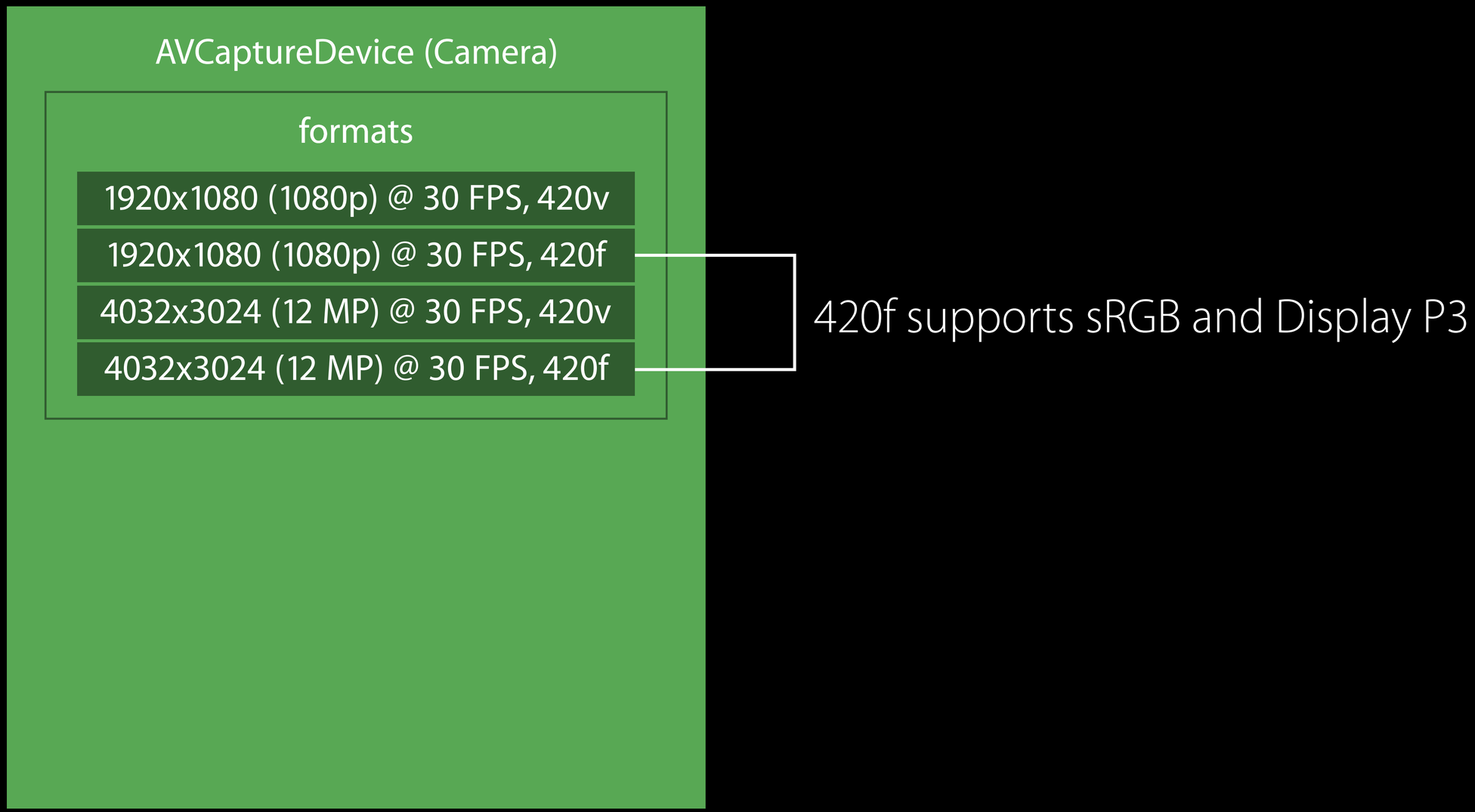
Capturing Display P3 Color on iPad Pro 9.7



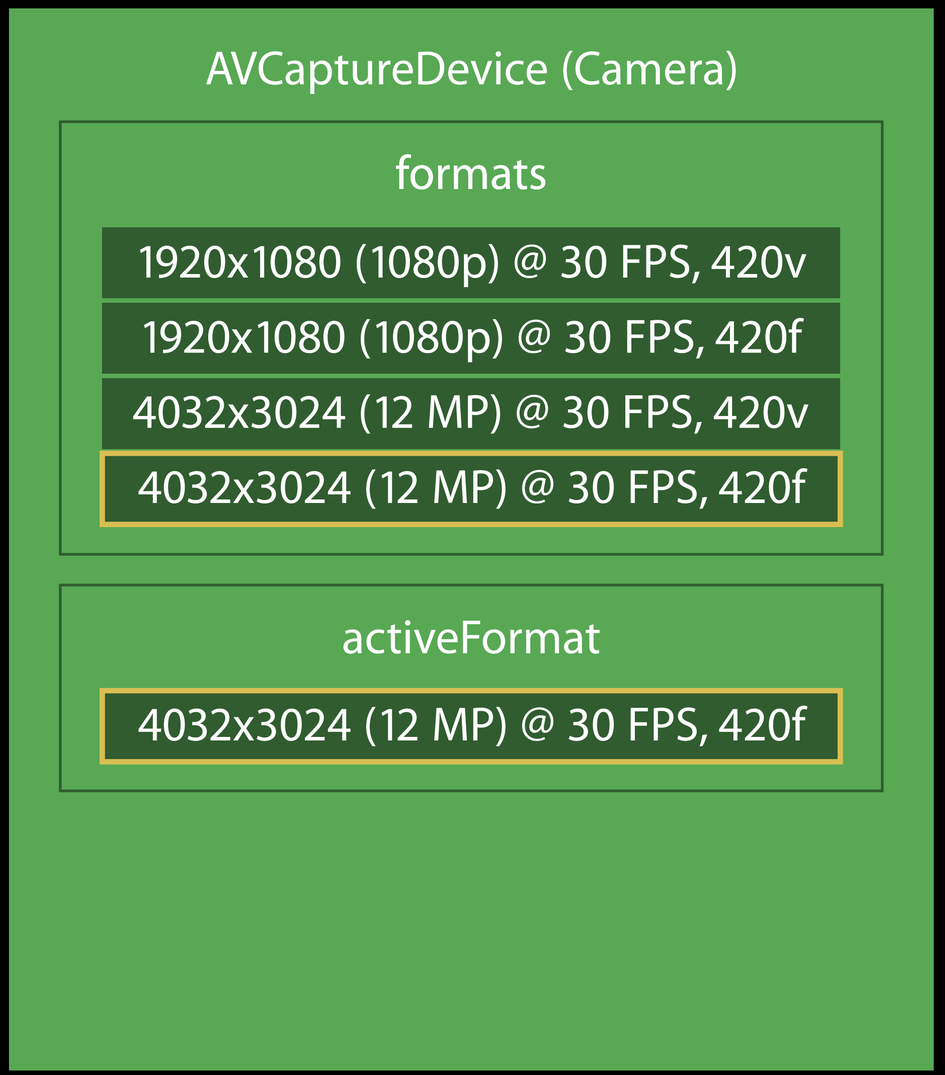
Capturing Display P3 Color on iPad Pro 9.7



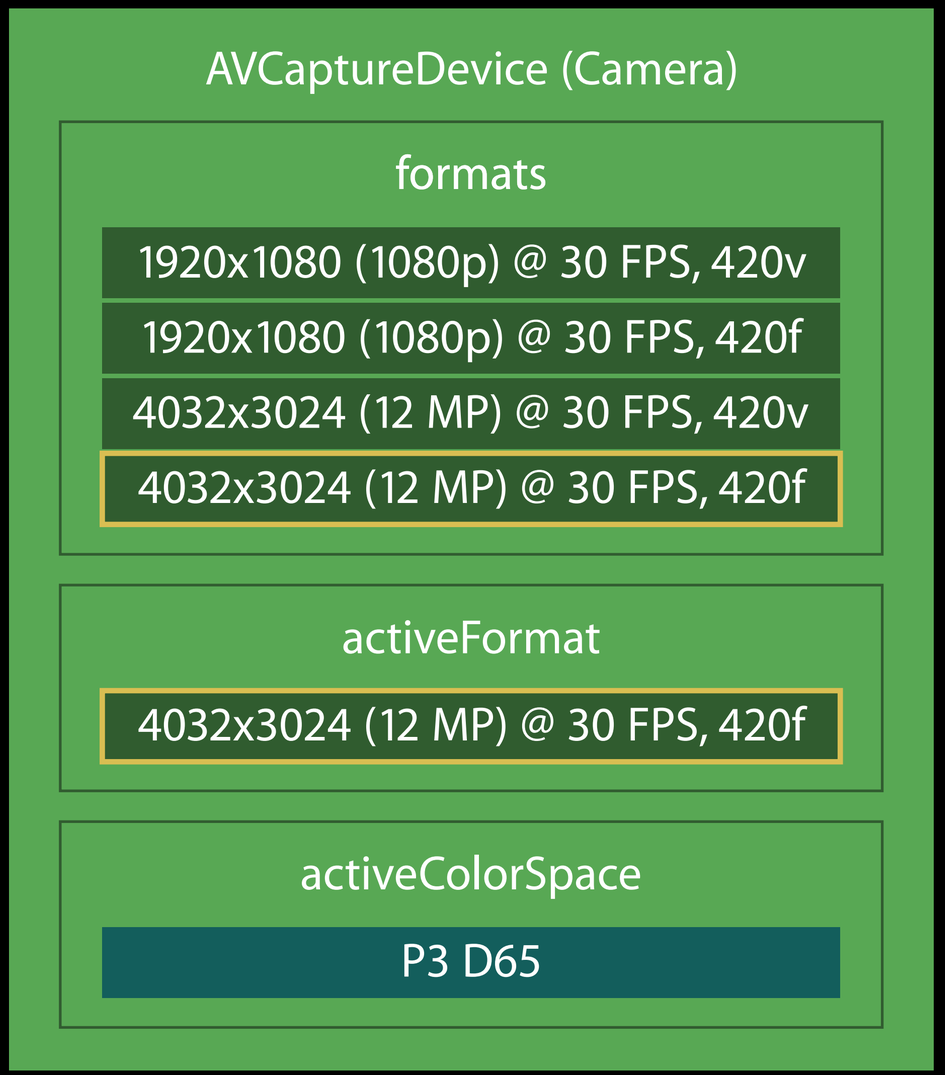
Capturing Display P3 Color on iPad Pro 9.7



Capturing Display P3 Color on iPad Pro 9.7



Capturing Display P3 Color on iPad Pro 9.7



Automatic Color Space Selection in AVCaptureSession

Automatic Color Space Selection in AVCaptureSession

```
public var automaticallyConfiguresCaptureDeviceForWideColor: Bool
```

Automatic Color Space Selection in AVCaptureSession

```
public var automaticallyConfiguresCaptureDeviceForWideColor: Bool
```

Session sets your device's `activeColorSpace` to `P3_D65` depending on your config

Automatic Color Space Selection in AVCaptureSession

```
public var automaticallyConfiguresCaptureDeviceForWideColor: Bool
```

Session sets your device's `activeColorSpace` to `P3_D65` depending on your config

AVCapturePhotoOutput must be present in the session

Automatic Color Space Selection in AVCaptureSession

Session contains photo output and…	Session configures device for…
AVCaptureVideoPreviewLayer	Display P3
AVCaptureMovieFileOutput	sRGB
AVCaptureVideoDataOutput	Display P3 *only if sessionPreset is Photo*

Forcing Display P3 Color

Forcing Display P3 Color

Set `session.automaticallyConfiguresDeviceForWideColor = false`

Forcing Display P3 Color

Set `session.automaticallyConfiguresDeviceForWideColor = false`

Set `device.activeFormat` to a format that supports wide color

Forcing Display P3 Color

Set `session.automaticallyConfiguresDeviceForWideColor = false`

Set `device.activeFormat` to a format that supports wide color

Set `device.activeColorSpace = P3_D65`

The Danger of Forcing Display P3 Color

The Danger of Forcing Display P3 Color

Display P3 is not well-supported in video

The Danger of Forcing Display P3 Color

Display P3 is not well-supported in video

VideoDataOutput callback should be color-aware and propagate color tags

The Danger of Forcing Display P3 Color

Display P3 is not well-supported in video

VideoDataOutput callback should be color-aware and propagate color tags

Display P3 movies from MovieFileOutput may render incorrectly on other platforms

Sharing Wide Color Photos

Sharing Wide Color Photos

Wide color JPEG files use a Display P3 Color Profile

Sharing Wide Color Photos

Wide color JPEG files use a Display P3 Color Profile

iCloud Photo Library is color-aware

Sharing Wide Color Photos

Wide color JPEG files use a Display P3 Color Profile

iCloud Photo Library is color-aware

Display P3 JPEGs via Messages and Mail are converted to **Apple Wide Color Sharing Profile**

Sharing Wide Color Photos

Wide color JPEG files use a Display P3 Color Profile

iCloud Photo Library is color-aware

Display P3 JPEGs via Messages and Mail are converted to **Apple Wide Color Sharing Profile**

Live Photo Editing and RAW Processing with Core Image	Nob Hill	Thursday 11:00AM
Working With Wide Color	Mission	Thursday 1:40PM

AVCapturePhotoOutput Wide Color Support on iPad Pro 9.7

AVCapturePhotoOutput Wide Color Support on iPad Pro 9.7

`‘420f’`, `‘BGRA’`, and `‘jpeg’`

AVCapturePhotoOutput Wide Color Support on iPad Pro 9.7

`‘420f’`, `‘BGRA’`, and `‘jpeg’`

Live Photos (still image and movie)

AVCapturePhotoOutput Wide Color Support on iPad Pro 9.7

`‘420f’`, `‘BGRA’`, and `‘jpeg’`

Live Photos (still image and movie)

Bracketed capture

Wide Color and RAW

Wide Color and RAW

Apple camera RAW images are inherently wide color

Wide Color and RAW

Apple camera RAW images are inherently wide color

Camera sensor primaries are rich enough to extract Display P3 or sRGB

More on Wide Color

Live Photo Editing and RAW Processing with Core Image	Nob Hill	Thursday 11:00AM
Working With Wide Color	Mission	Thursday 1:40PM

Summary

Use `AVCapturePhotoOutput` for improved usability

Summary

Use `AVCapturePhotoOutput` for improved usability

- Live Photos

Summary

Use `AVCapturePhotoOutput` for improved usability

- Live Photos
- RAW, RAW + JPEG and DNG

Summary

Use `AVCapturePhotoOutput` for improved usability

- Live Photos
- RAW, RAW + JPEG and DNG
- Preview Images

Summary

Use `AVCapturePhotoOutput` for improved usability

- Live Photos
- RAW, RAW + JPEG and DNG
- Preview Images
- Wide Color Photos

AVCapturePhotoOutput—Beyond the Basics

Session 511: A chalk talk addendum

AVCapturePhotoOutput—Beyond the Basics

Session 511: A chalk talk addendum

Scene monitoring in `AVCapturePhotoOutput`

AVCapturePhotoOutput—Beyond the Basics

Session 511: A chalk talk addendum

Scene monitoring in `AVCapturePhotoOutput`

Resource preparation and reclamation in `AVCapturePhotoOutput`

AVCapturePhotoOutput—Beyond the Basics

Session 511: A chalk talk addendum

Scene monitoring in `AVCapturePhotoOutput`

Resource preparation and reclamation in `AVCapturePhotoOutput`

Changes to camera privacy policy in iOS 10

More Information

https://developer.apple.com/wwdc16/501

Related Sessions

AVCapturePhotoOutput—Beyond the Basics	Video	At your leisure
Live Photo Editing and RAW Processing with Core Image	Nob Hill	Thursday 11:00AM
Working With Wide Color	Mission	Thursday 1:40PM

Labs

Photo Capture Lab	Graphics, Games, and Media Lab D	Tuesday 1:00PM
PhotoKit Lab	Graphics, Games, and Media Lab D	Tuesday 4:00PM
Color Lab	Frameworks Lab A	Wednesday 1:00PM
Photo Capture Lab	Graphics, Games, and Media Lab C	Thursday 9:00AM
Live Photo & Core Image Lab	Graphics, Games, and Media Lab C	Thursday 1:30PM
Live Photo & Core Image Lab	Graphics, Games, and Media Lab D	Friday 9:00AM
Color Lab	Graphics, Games, and Media Lab C	Friday 4:00PM

WWDC16