

[AVFoundation](#) / [AVCaptureSession](#)

Class

AVCaptureSession

An object that configures capture behavior and coordinates the flow of data from input devices to capture outputs.

iOS 4.0+ | iPadOS 4.0+ | Mac Catalyst 14.0+ | macOS 10.7+ | tvOS 17.0+ | visionOS 1.0+

```
class AVCaptureSession
```

Mentioned in

-  Enhancing your app experience with the Camera Control
-  Setting up a capture session

Overview

To perform real-time capture, you instantiate a capture session and add appropriate inputs and outputs. The following code fragment illustrates how to configure a capture device to record audio.

```
// Create the capture session.  
let captureSession = AVCaptureSession()  
  
// Find the default audio device.  
guard let audioDevice = AVCaptureDevice.default(for: .audio) else { return }  
  
do {  
    // Wrap the audio device in a capture device input.  
    let audioInput = try AVCaptureDeviceInput(device: audioDevice)  
    // If the input can be added, add it to the session.  
}
```

```
if captureSession.canAddInput(audioInput) {  
    captureSession.addInput(audioInput)  
}  
}  
} catch {  
    // Configuration failed. Handle error.  
}
```

Call the [startRunning\(\)](#) method to start the flow of data from the inputs to the outputs, and call the [stopRunning\(\)](#) method to stop the flow.

Important

The [startRunning\(\)](#) method is a blocking call which can take some time, therefore start the session on a serial dispatch queue so that you don't block the main queue (which keeps the UI responsive). See [AVCam: Building a camera app](#) for an implementation example.

You use the [sessionPreset](#) property to customize the quality level, bitrate, or other settings for the output. Most common capture configurations are available through session presets; however, some specialized options (such as high frame rate) require directly setting a capture format on an [AVCaptureDevice](#) instance.

Topics

Configuring a session

`func beginConfiguration()`

Marks the beginning of changes to a running capture session's configuration to perform in a single atomic update.

`func commitConfiguration()`

Commits one or more changes to a running capture session's configuration in a single atomic update.

Setting a session preset

`struct Preset`

Presets that define standard configurations for a capture session.

`func canSetSessionPreset(AVCaptureSession.Preset) -> Bool`

Determines whether you can configure a capture session with the specified preset.

```
var sessionPreset: AVCaptureSession.Preset
```

A preset value that indicates the quality level or bit rate of the output.

Configuring inputs

```
var inputs: [AVCaptureInput]
```

The inputs that provide media data to a capture session.

```
func canAddInput(AVCaptureInput) -> Bool
```

Determines whether you can add an input to a session.

```
func addInput(AVCaptureInput)
```

Adds a capture input to the session.

```
func removeInput(AVCaptureInput)
```

Removes an input from the session.

Configuring outputs

```
var outputs: [AVCaptureOutput]
```

The output destinations to which a captures session sends its data.

```
func canAddOutput(AVCaptureOutput) -> Bool
```

Determines whether you can add an output to a session.

```
func addOutput(AVCaptureOutput)
```

Adds an output to the capture session.

```
func removeOutput(AVCaptureOutput)
```

Removes an output from a capture session.

Connecting inputs and outputs

```
var connections: [AVCaptureConnection]
```

The connections between inputs and outputs that a capture session contains.

```
func addConnection(AVCaptureConnection)
```

Adds a connection to the capture session.

```
func canAddConnection(AVCaptureConnection) -> Bool
```

Determines whether you can add a connection to a capture session.

```
func addInputWithNoConnections( AVCaptureInput )
```

Adds a capture input to a session without forming any connections.

```
func addOutputWithNoConnections( AVCaptureOutput )
```

Adds a capture output to the session without forming any connections.

```
func removeConnection( AVCaptureConnection )
```

Removes a capture connection from the session.

```
class AVCaptureAudioChannel
```

An object that monitors average and peak power levels for an audio channel in a capture connection.

Configuring deferred start

```
var isManualDeferredStartSupported: Bool
```

A BOOL value that indicates whether the session supports manually running deferred start.

```
var automaticallyRunsDeferredStart: Bool
```

A Boolean value that indicates whether deferred start runs automatically.

```
func runDeferredStartWhenNeeded()
```

Tells the session to run deferred start when appropriate.

```
var deferredStartDelegate: (any AVCaptureSessionDeferredStartDelegate)?
```

A delegate object that observes events about deferred start.

```
var deferredStartDelegateCallbackQueue: dispatch_queue_t?
```

The dispatch queue on which the session calls deferred start delegate methods.

```
func setDeferredStartDelegate((any AVCaptureSessionDeferredStartDelegate)?, deferredStartDelegateCallbackQueue: dispatch_queue_t?)
```

Sets a delegate object for the session to call when performing deferred start.

```
protocol AVCaptureSessionDeferredStartDelegate
```

A protocol that defines the interface to respond to events about a capture session's deferred start.

Configuring capture controls

```
var supportsControls: Bool
```

A Boolean value that indicates whether a capture session supports controls.

```
var maxControlsCount: Int
```

The maximum number of controls a capture session supports.

```
var controls: [AVCaptureControl]
```

The controls that allow configuring the camera system from device hardware.

```
func canAddControl(AVCaptureControl) -> Bool
```

Returns a Boolean value that indicates whether a capture session add the specified control.

```
func addControl(AVCaptureControl)
```

Adds a control to a capture session.

```
func removeControl(AVCaptureControl)
```

Removes a control from a capture session.

```
func setControlsDelegate((any AVCaptureSessionControlsDelegate)?, queue : dispatch_queue_t?)
```

Sets a delegate object for the system to call when it activates and presents controls.

```
protocol AVCaptureSessionControlsDelegate
```

A protocol that defines the interface to respond to capture control activation and presentation events.

```
var controlsDelegate: (any AVCaptureSessionControlsDelegate)?
```

A delegate object that observes changes to the state of capture controls.

```
var controlsDelegateCallbackQueue: dispatch_queue_t?
```

The dispatch queue on which the system calls controls delegate methods.

Managing the session life cycle

```
func startRunning()
```

Starts the flow of data through the capture pipeline.

```
func stopRunning()
```

Stops the flow of data through the capture pipeline.

Observing session state

```
var isRunning: Bool
```

A Boolean value that indicates whether the capture session is in a running state.

```
var isInterrupted: Bool
```

A Boolean value that indicates whether the capture session is in an interrupted state.

```
class let didStartRunningNotification: NSNotification.Name
```

A notification the system posts when a capture session starts.

```
class let didStopRunningNotification: NSNotification.Name
```

A notification the system posts when a capture session stops.

```
class let wasInterruptedNotification: NSNotification.Name
```

A notification the system posts when it interrupts a capture session.

```
class let interruptionEndedNotification: NSNotification.Name
```

A notification the system posts when an interruption to a capture session finishes.

```
class let runtimeErrorNotification: NSNotification.Name
```

A notification the system posts when an error occurs during a capture session.

Configuring multitasking

```
var isMultitaskingCameraAccessSupported: Bool
```

A Boolean value that indicates whether the capture session supports using the camera while multitasking.

```
var isMultitaskingCameraAccessEnabled: Bool
```

A Boolean value that indicates whether the capture session enables access to the camera while multitasking.

Monitoring performance

```
var hardwareCost: Float
```

A value that indicates the percentage of the session's available hardware budget in use.

Configuring the app's audio session

```
var usesApplicationAudioSession: Bool
```

A Boolean value that indicates whether the capture session uses the app's shared audio session.

```
var automaticallyConfiguresApplicationAudioSession: Bool
```

A Boolean value that indicates whether the capture session automatically changes settings in the app's shared audio session.

```
var configuresApplicationAudioSessionToMixWithOthers: Bool
```

A Boolean value that indicates whether the capture session configures the app's audio session to mix with others.

```
var configuresApplicationAudioSessionForBluetoothHighQualityRecording: Bool
```

A Boolean value that indicates whether the capture session configures the app's audio session for bluetooth high-quality recording.

Managing color spaces

```
var automaticallyConfiguresCaptureDeviceForWideColor: Bool
```

A Boolean value that specifies whether the session should automatically use wide-gamut color where available.

Synchronizing output

```
var synchronizationClock: CMClock?
```

A clock to use for output synchronization.

```
var masterClock: CMClock?
```

A clock object used for output synchronization.

Deprecated

Relationships

Inherits From

NSObject

Inherited By

Conforms To

CVarArg
CustomDebugStringConvertible
CustomStringConvertible
Equatable
Hashable
NSObjectProtocol

See Also

Capture sessions

- 📄 Setting up a capture session
 - Configure input devices, output media, preview views, and basic settings before capturing photos or video.
- 📄 Accessing the camera while multitasking on iPad
 - Operate the camera in Split View, Slide Over, Picture in Picture, and Stage Manager modes.
- { } AVCam: Building a camera app
 - Capture photos and record video using the front and rear iPhone and iPad cameras.
- { } Capturing Cinematic video
 - Capture video with an adjustable depth of field and focus points.
- { } AVMultiCamPiP: Capturing from Multiple Cameras
 - Simultaneously record the output from the front and back cameras into a single movie file by using a multi-camera capture session.
- { } AVCamBarcode: detecting barcodes and faces
 - Identify machine readable codes or faces by using the camera.

class AVCaptureMultiCamSession

A capture session that supports simultaneous capture from multiple inputs of the same media type.

```
class AVCaptureInput
```

An abstract superclass for objects that provide input data to a capture session.

```
class AVCaptureOutput
```

An abstract superclass for objects that provide media output destinations for a capture session.

```
class AVCaptureConnection
```

An object that represents a connection from a capture input to a capture output.