Delivering Intuitive Media Playback with AVKit

Jed Lewison, AVKit Engineer
Dan Wright, AVKit Engineer
Media playback with AVKit
What’s new
Best practices
AVKit in the Stack

AVKit

AVFoundation

CoreMedia
AVKit in the Stack

- AVKit
- AVFoundation
- CoreMedia
- UIKit
- AVPlayerViewController
AVKit in the Stack

AVKit

AVFoundation

CoreMedia

UIKit

AVPlayerViewController

AVPlayerView

AppKit

AVRoutePickerView
// Media Playback with AVPlayerViewController on iOS and tvOS

import AVKit

// 1) Create an AVPlayer
let player = AVPlayer(url: "https://my.example/video.m3u8")

// 2) Create an AVPlayerViewController
let playerViewController = AVPlayerViewController()
playerViewController.player = player

// 3) Show it
present(playerViewController, animated: true)
import AVKit

// 1) Create an AVPlayer
let player = AVPlayer(url: "https://my.example/video.m3u8")

// 2) Create an AVPlayerViewController
let playerViewController = AVPlayerViewController()
playerViewController.player = player

// 3) Show it
present(playerViewController, animated: true)
import AVKit

// 1) Create an AVPlayer
let player = AVPlayer(url: "https://my.example/video.m3u8")

// 2) Create an AVPlayerViewController
let playerViewController = AVPlayerViewController()
playerViewController.player = player

// 3) Show it
present(playerViewController, animated: true)
// Media Playback with AVPlayerViewController on iOS and tvOS

import AVKit

// 1) Create an AVPlayer
let player = AVPlayer(url: "https://my.example/video.m3u8")

// 2) Create an AVPlayerViewController
let playerViewController = AVPlayerViewController()
playerViewController.player = player

// 3) Show it
present(playerViewController, animated: true)
Media Playback with AVKit

- HLS
- offline HLS
- file-based playback
- network playback
- content key management
- HDR
- 4K
- AirPlay 2
- external display support
- background audio
- movie compositions
- AVMovie
- AVQueuePlayer
- AVComposition
- AVMutableMovie
- AVPlayerItem
- AVAudioMix
- movie compositions
- AVAudioSession
- AVPlayerLayer
- AVAsset
- AVURLAsset
- AVAssetTrack
- AVMutableAssetTrack
- AVMovie
- AVMetadataItem
- AVMediaSelectionOption
- subtitles
- background audio
- temporal compositions
- AVAssetTrack
- AVMutableAssetTrack
- AVMetadataItem
- file-based playback
- network playback
AVPlayerViewController on iOS

What’s new for iOS 13
AVPlayerViewController

What’s new for iOS 13

Full screen callbacks

AVPlayerViewController in iPad apps on the Mac

External metadata

Improved support for custom controls
AVKit on tvOS
WWDC 2016
20 minutes remaining

AVKit on tvOS offers an intuitive user interface for interaction with your media on this powerful new platform. AVKit is built on AVFoundation, which gives you access to the full potential of platform services for operating on time-based audiovisual media. Learn how to leverage the new content proposal APIs, and how to create a seamless interstitial playback experience. Management of media metadata and chapter navigation will also be covered.
AVKit on tvOS

WWDC 2016
20 minutes remaining

AVKit on tvOS offers an intuitive user interface for interaction with your media on this powerful new platform. AVKit is built on AVFoundation, which gives you access to the full potential of platform services for operating on time-based audiovisual media. Learn how to leverage the new content proposal APIs, and how to create a seamless interstitial playback experience. Management of media metadata and chapter navigation will also be covered.
Full Screen Callbacks

Extends AVPlayerViewControllerDelegate
Full Screen Callbacks

Extends AVPlayerViewControllerDelegate

Notifies delegate when beginning or ending full screen presentation
Full Screen Callbacks

Extends AVPlayerViewControllerDelegate

Notifies delegate when beginning or ending full screen presentation

```
@available(iOS 12.0, *)

func playerViewController(_ playerViewController: AVPlayerViewController, 
willBeginFullScreenPresentationWithAnimationCoordinator coordinator: 
UIViewControllerTransitionCoordinator)

func playerViewController(_ playerViewController: AVPlayerViewController, 
willEndFullScreenPresentationWithAnimationCoordinator coordinator: 
UIViewControllerTransitionCoordinator)
```
Full Screen Callbacks

Extends AVPlayerViewControllerDelegate

Notifies delegate when beginning or ending full screen presentation

```
@available(iOS 12.0, *)

func playerViewController(_ playerViewController: AVPlayerViewController, 
willBeginFullScreenPresentationWithAnimationCoordinator coordinator: 
UIViewControllerTransitionCoordinator coordinator: 
UIViewControllerTransitionCoordinator)

func playerViewController(_ playerViewController: AVPlayerViewController, 
willEndFullScreenPresentationWithAnimationCoordinator coordinator: 
UIViewControllerTransitionCoordinator coordinator: 
UIViewControllerTransitionCoordinator)
```
Full Screen Callbacks

Extends AVPlayerViewControllerDelegate

Notifies delegate when beginning or ending full screen presentation

@available(iOS 12.0, *)

func playerViewController(_ playerViewController: AVPlayerViewController,
    willBeginFullScreenPresentationWithAnimationCoordinator coordinator:
    UIViewControllerTransitionCoordinator)

func playerViewController(_ playerViewController: AVPlayerViewController,
    willEndFullScreenPresentationWithAnimationCoordinator coordinator:
    UIViewControllerTransitionCoordinator)
// Implementing AVPlayerViewControllerDelegate's full screen callbacks

func playerViewController(
    _ playerViewController: AVPlayerViewController,
    willBeginFullScreenPresentationWithAnimationCoordinator coordinator: UIViewControllerTransitionCoordinator
) {
    coordinator.animate(alongsideTransition: { (context) in
        // Add coordinated animations
    }) { (context) in
        if context.isCancelled {
            // Still embedded inline
        } else {
            // Presented full screen
            // Take strong reference to playerViewController if needed
        }
    }
}
// Implementing AVPlayerViewControllerDelegate's full screen callbacks

func playerViewController(
    _ playerViewController: AVPlayerViewController,
    willBeginFullScreenPresentationWithAnimationCoordinator coordinator: UIViewControllerTransitionCoordinator
) {
    coordinator.animate(alongsideTransition: { (context) in
        // Add coordinated animations
    }) { (context) in
        if context.isCancelled {
            // Still embedded inline
        } else {
            // Presented full screen
            // Take strong reference to playerViewController if needed
        }
    }
}
Full Screen Callbacks
Retargeting dismissal animation
Full Screen Callbacks
Retargeting dismissal animation

Keep embedded player view controller alive when full screen
• Deallocating will stop full screen playback
• Okay to be offscreen or removed from superview / parent
Full Screen Callbacks

Retargeting dismissal animation

Keep embedded player view controller alive when full screen
• Deallocating will stop full screen playback
• Okay to be offscreen or removed from superview / parent

Use delegate to restore playerViewController or retarget animation

```swift
func playerViewController(
    _ playerViewController: AVPlayerViewController,
    willEndFullScreenPresentationWithAnimationCoordinator coordinator: UIViewControllerTransitionCoordinator
) {
    // If scrolled away, update playerViewController’s layout here
}
```
Full Screen Callbacks

Track full screen presentation state with AVPlayerViewControllerDelegate

• Do not rely on viewWillAppear(_: _) and friends

Works for both embedded and full screen presentations

Keep embedded player view controller alive when full screen

Can retarget dismissal animation when embedding inline
How many new lines of code?
AVPlayerViewController in iPad Apps on the Mac

Exact same API as on iOS

Picture-in-picture support
• Includes AVPictureInPictureController (also for AppKit)
• And AVPlayerView, for AppKit-based apps

Touch Bar, keyboard, and Now Playing support

Audio and AirPlay routing

Available in macOS 10.15
External Metadata
External Metadata

Supplement missing metadata

• Title, artwork, and more

Same API as tvOS:

```swift
@available(iOS 12.0, *)
extension AVPlayerItem {
    open var externalMetadata: [AVMetadataItem]
}
```
Improved Support for Custom Controls

Interactive dismissals
Landscape support for portrait-only apps
Keyboard and Touch Bar support
Now Playing management
Automatic video zoom
Custom Playback Controls

The basics

Set `showsPlaybackControls` to `false`
Custom Playback Controls

The basics

Set `showsPlaybackControls` to `false`

Present modally
Custom Playback Controls

The basics

Set `showsPlaybackControls` to `false`

Present modally

Add controls to `contentOverlayView`
Custom Playback Controls
The basics

Set `showsPlaybackControls` to `false`

Present modally

Add controls to `contentOverlayView`

For the best user experience, you should:

• Override `UIViewController` methods for status bar, home indicator

• Pass unhandled touches through your view

• Let AVKit handle double-tap for video zoom
What’s New

Full screen callbacks

AVPlayerViewController in iPad apps for the Mac
• Plus picture-in-picture support for AppKit-based apps

External metadata

Improved support for custom controls
AVPlayerViewController on iOS

Best practices
AVPlayerViewController on iOS: Best Practices

• Showing full screen
• Embedding inline
• Picture-in-picture
Showing Full Screen

Covers UIWindowScene coordinate space
Showing Full Screen

Covers **UIWindowScene** coordinate space

Use cases:

- Splash screen
- Full screen playback
Splash Screen
Requirements

Underneath your UI
No interactive playback controls
No need to hide status bar or home indicator
Possibly looping
Video should always fill screen
For video with alpha, custom background color
Audio is secondary
Splash Screen
UIViewController containment API

Add as child:
parent.addChild(playerViewController)
p(parent).view.addSubview(playerViewController.view) // Or other UIView insertion API
// Enable auto layout and set up constraints
playerViewController.didMove(toParent: parent)

Remove from parent:
playerViewController.willMove(toParent: nil)
playerViewController.view.removeFromSuperview()
playerViewController.removeFromParent()
Splash Screen
Configure AVPlayerViewController

Disable playback controls:
```swift
playerViewController.showsPlaybackControls = false
```

Make video fill entire screen:
```swift
playerViewController.videoGravity = .resizeAspectFill
```

Set background color (if needed):
```swift
playerViewController.view.backgroundColor = .clear // Or any other UIColor
```
Splash Screen
AVFoundation best practices

Disable `AVPlayer.allowsExternalPlayback`
Splash Screen
AVFoundation best practices

Disable `AVPlayer.allowsExternalPlayback`

Configure `AVAudioSession` for secondary media playback
• Use `.ambient` category
Splash Screen
AVFoundation best practices

Disable `AVPlayer.allowsExternalPlayback`

Configure `AVAudioSession` for secondary media playback
• Use `.ambient category`

Observe `AVAudioSession.silenceSecondaryAudioHintNotification`
• Honor `AVAudioSession.secondaryAudioShouldBeSilencedHint`
Full Screen Playback
AVPlayerViewController best practices

Landscape
Dark Mode
Wireless Routing
External Playback
Keyboard
Subtitles
Full Screen
Volume
Rotate
Hide Status Bar
White Point
Home Indicator
Device-aware video zoom
Accessibility
Animated Mute
Interactive dismissals
Picture-in-picture
Loading Indicator
Media Selection
Control Center
Now Playing
External Playback
Touch Bar
Full Screen
Volume
Rotate
Captions
Home Indicator
Device-aware video zoom
Accessibility
Animated Mute
Interactive dismissals
Picture-in-picture
Loading Indicator
Media Selection
Control Center
Now Playing
// Media Playback with AVPlayerViewController on iOS and tvOS

import AVKit

// 1a) Create an AVPlayer
let player = AVPlayer(url: "https://my.example/video.m3u8")

// 1b) Add external metadata if needed
player.currentItem?.externalMetadata = // Array of [AVMetadataItem]

// 2) Create an AVPlayerViewController
let playerViewController = AVPlayerViewController()
playerViewController.player = player

// 3) Show it
present(playerViewController, animated: true)
Full Screen Playback
AVPlayerViewController best practices

Present modally
Full Screen Playback
AVPlayerViewController best practices

Present modally

Use the default `modalPresentationStyle`
Full Screen Playback
AVPlayerViewController best practices

Present modally

Use the default `modalPresentationStyle`

Do not set `AVPlayerViewController.videoGravity`
Full Screen Playback
AVPlayerViewController best practices

Present modally

Use the default `modalPresentationStyle`

Do not set `AVPlayerViewController.videoGravity`

Use `AVPlayerViewControllerDelegate` to track full screen presentation state

• Do not rely on `viewWillAppear(_:)` and friends
Set up `AVPlayerItem` before buffering

- Set `AVPlayer.rate` for setting item
Full Screen Playback
AVFoundation best practices

Set up AVPlayerItem before buffering
• Set AVPlayer.rate for setting item

Observe AVPlayer.status and AVPlayerItem.status
• Don’t begin playback until status is .readyToPlay
• Check error property when status is .failed
  - Rebuild AVFoundation objects if .mediaServicesWereReset
Set up `AVPlayerItem` before buffering

• Set `AVPlayer.rate` for setting item

Observe `AVPlayer.status` and `AVPlayerItem.status`

• Don’t begin playback until status is `.readyToPlay`

• Check `error` property when status is `.failed`
  - Rebuild AVFoundation objects if `.mediaServicesWereReset`

Enable `AVPlayer.usesExternalPlaybackWhileExternalScreenIsActive`
Full Screen Playback
AVFoundation best practices

Set up `AVPlayerItem` before buffering

• Set `AVPlayer.rate` for setting item

Observe `AVPlayer.status` and `AVPlayerItem.status`

• Don’t begin playback until status is `.readyToPlay`

• Check `error` property when status is `.failed`
  - Rebuild AVFoundation objects if `.mediaServicesWereReset`

Enable `AVPlayer.usesExternalPlaybackWhileExternalScreenIsActive`

Configure `AVAudioSession` for `.playback`
What's New in HTTP Live Streaming

WWDC 2016
24 minutes remaining

HTTP Live Streaming offers a reliable media playback experiences over a wide variety of network conditions. Join us to learn how to deliver offline playback using the same media assets you already host for online playback. See new additions to the HLS protocol, such as support for fragmented MP4 and in-playlist metadata.
Embedding AVPlayerViewController Inline

Best practices

Be prepared to begin and end full screen playback
- Track state using AVPlayerViewControllerDelegate
- Keep strong reference to embedded AVPlayerViewController when full screen
- Retarget dismissal animations when ending full screen presentation
Embedding AVPlayerViewController Inline

Best practices

Be prepared to begin and end full screen playback

- Track state using AVPlayerViewControllerDelegate
- Keep strong reference to embedded AVPlayerViewController when full screen
- Retarget dismissal animations when ending full screen presentation

Leave modalPresentationStyle as .fullScreen
Embedding AVPlayerViewController Inline

Best practices

Can style embedded content without impacting full screen:

- `videoGravity`
- `view.layer.cornerRadius`, `cornerCurve`, and `maskedCorners`
- `view.backgroundColor`
Embedding AVPlayerViewController Inline

Best practices

Can style embedded content without impacting full screen:

- `videoGravity`
- `view.layer.cornerRadius`, `cornerCurve`, and `maskedCorners`
- `view.backgroundColor`

For automatically entering and exiting full screen:

- `entersFullScreenWhenPlaybackBegins`
- `exitsFullScreenWhenPlaybackEnds`
Embedding AVPlayerViewController Inline

Best practices

Can style embedded content without impacting full screen:
- `videoGravity`
- `view.layer.cornerRadius`, `cornerCurve`, and `maskedCorners`
- `view.backgroundColor`

For automatically entering and exiting full screen:
- `entersFullScreenWhenPlaybackBegins`
- `exitsFullScreenWhenPlaybackEnds`

Adopt UIViewController containment API
Embedding AVPlayerViewController Inline
UIViewController containment API

Add as child:

```swift
parent.addChild(playerViewController)
containerView.addSubview(playerViewController.view) // Don’t put views on top
// Enable auto layout and set up constraints
playerViewController.didMove(toParent: parent)
```

Remove from parent:

```swift
playerViewController.willMove(toParent: nil)
playerViewController.view.removeFromSuperview()
playerViewController.removeFromParent()
```
Embedding AVPlayerViewController Inline
Showing poster images

Use `.contentOverlayView`

• Okay to use before setting player or player item
Embedding AVPlayerViewController Inline
Showing poster images

Use `.contentOverlayView`
• Okay to use before setting player or player item

Observe `.isReadyForDisplay`
• Returns `true` when first frame is ready
Embedding AVPlayerViewController Inline

Showing poster images

Use `.contentOverlayView`
• Okay to use before setting player or player item

Observe `.isReadyForDisplay`
• Returns `true` when first frame is ready

```swift
let token = pvc.observe(.isReadyForDisplay, options: [.initial]) {
    [weak self] observed, _ in
    if observed.isReadyForDisplay {
        // Hide any poster frame or placeholder
    }
}
```
Picture-in-picture
Configuring your app

Configure AVAudioSession for .playback

Add background audio mode entitlement in Xcode
Picture-in-picture
Continue playing when entering background

Don’t pause playback when entering background
• Picture-in-picture may be starting!
Picture-in-picture
Continue playing when entering background

Don’t pause playback when entering background
• Picture-in-picture may be starting!

If you must pause, wait until in background
• Only if picture-in-picture hasn’t started
• Pre-iOS 13: UIApplicationStateBackground
• iOS 13 and later:UISceneActivationStateBackground
Picture-in-picture
AVPlayerViewController best practices

Track state using AVPlayerViewControllerDelegate
Picture-in-picture
AVPlayerViewController best practices

Track state using AVPlayerViewControllerDelegate

Application can stop picture-in-picture
• Toggle AVPlayerViewController.allowsPictureInPicturePlayback
Picture-in-picture
AVPlayerViewController best practices

Track state using AVPlayerViewControllerDelegate

Application can stop picture-in-picture
• Toggle AVPlayerViewController.allowsPictureInPicturePlayback

Be prepared for dismissal when starting
• AVKit prevents deallocation while active
Picture-in-picture
AVPlayerViewController best practices

Track state using `AVPlayerViewControllerDelegate`

Application can stop picture-in-picture
• Toggle `AVPlayerViewController.allowsPictureInPicturePlayback`

Be prepared for dismissal when starting
• AVKit prevents deallocation while active

Always restore user interface when requested by user
func playerViewController(
    _ playerViewController: AVPlayerViewController,
    restoreUserInterfaceForPictureInPictureStopWithCompletionHandler completionHandler: @escaping (Bool) -> Void)
{
    presentingViewController.present(playerViewController, animated: true) {
        // Must invoke completionHandler
        completionHandler(true)
    }
}
AVPlayerViewController on iOS

Best practices

Sample code

• developer.apple.com/wwdc19/503
AVPlayerViewController: What’s New for tvOS

Dan Wright, AVKit Engineer
What AVPlayerViewController Provides on tvOS

Full screen interactive playback

Provides standard controls for navigation

Supports a wide variety of remote controllers

Updates Now Playing information

Interstitials

Content proposals
What’s New in AVKit for tvOS 13

• Updated appearance and fine-precision scrubbing (12.3)
• Custom interactive overlays
• Live broadcast channel-flipping
• Parental content restriction enforcement
What’s New in AVKit for tvOS 13

• Updated appearance and fine-precision scrubbing (12.3)
• Custom interactive overlays
• Live broadcast channel-flipping
• Parental content restriction enforcement
What’s New in AVKit for tvOS 13

Updated appearance and fine-precision scrubbing (12.3)

Custom interactive overlays

Live broadcast channel-flipping

Parental content restriction enforcement
Custom Overlays

Normally hidden
Custom Overlays

Normally hidden

On-screen hint guides the user
Custom Overlays

Normally hidden

On-screen hint guides the user

Revealed when the user swipes up
Custom Overlays

Normally hidden

On-screen hint guides the user

Revealed when the user swipes up

Fully interactive
Custom Overlays

Normally hidden

On-screen hint guides the user

Revealed when the user swipes up

Fully interactive

Defined by your view controller
Custom Overlays

Normally hidden

On-screen hint guides the user

Revealed when the user swipes up

Fully interactive

Defined by your view controller

Use `contentOverlayView` for persistent, non-interactive elements
Custom Interactive Overlay

Set the `customOverlayViewController` property of the player view controller.

```swift
let customInteractiveVideoOverlay = UIViewController(nibName: "CustomInteractiveVideoOverlay", bundle: nil)
newPlayerViewController.customOverlayViewController = customInteractiveVideoOverlay
```
What’s New in AVKit for tvOS 13

- Updated appearance and fine-precision scrubbing (12.3)
- Custom interactive overlays
- Live broadcast channel-flipping
- Parental content restriction enforcement
Channel Flipping

For live streams
Channel Flipping

For live streams

Swipe horizontally for next/previous channels
Channel Flipping

For live streams

Swipe horizontally for next/previous channels

Channel interstitial screen describes a channel while it loads
// Extending the delegate to supporting channel-flipping:

func playerViewController(_ playerViewController: AVPlayerViewController, skipToNextChannel completion: @escaping (Bool) -> Void) {
    // Load content for the new channel, and update the playerViewController.
    if let url = URL(string: "https://example.com/videos/a_channel/master.m3u8") {
        let playerItem = AVPlayerItem(url: url)
        playerViewController.player?.replaceCurrentItem(with: playerItem)
        completion(true) // Indicate successful completion.
    } else {
        completion(false) // Indicate failure.
    }
}

func playerViewController(_ playerViewController: AVPlayerViewController, skipToPreviousChannel completion: @escaping (Bool) -> Void) {
}
// Extending the delegate to supporting channel-flipping:

```swift
func nextChannelInterstitialViewController(for playerViewController: AVPlayerViewController) -> UIViewController {
    let channelInterstitial = MyChannelInterstitialViewController()
    // Update channelInterstitial with information about next channel before returning!
    return channelInterstitial
}
```

```swift
func previousChannelInterstitialViewController(for playerViewController: AVPlayerViewController) -> UIViewController {
    // Similar to above, except for previous channel.
}
```
What’s New in AVKit for tvOS 13

Updated appearance and fine-precision scrubbing (12.3)
Custom interactive overlays
Live broadcast channel-flipping
Parental content restriction enforcement
Parental Content Restrictions

Specify the media content rating
Parental Content Restrictions

Specify the media content rating

Provide the rating at the start
Parental Content Restrictions

Specify the media content rating

Provide the rating at the start

If restricted by passcode, the user will be prompted
Parental Content Restrictions

Specify the media content rating

Provide the rating at the start

If restricted by passcode, the user will be prompted

Applications can request access earlier
Parental Content Restrictions

Specify the media content rating

Provide the rating at the start

If restricted by passcode, the user will be prompted

Applications can request access earlier

Failure dismisses player view controller
Parental Content Restrictions

Specify the media content rating

Provide the rating at the start

If restricted by passcode, the user will be prompted

Applications can request access earlier

Failure dismisses player view controller

Custom playback user interfaces can also request access to restricted content
// Providing media content rating metadata

func metadataItemForMediaContentRating(_ rating: String) -> AVMetadataItem {
    let metadataItem = AVMutableMetadataItem()
    metadataItem.value = value as NSString
    metadataItem.extendedLanguageTag = "und"
    metadataItem.identifier = .iTunesMetadataContentRating
    return metadataItem
}

let playerItem = AVPlayerItem(url: url)
// World-wide content ratings may be used; the user’s rating region will be preferred.
playerItem.externalMetadata = [metadataItemForMediaContentRating("PG-13"), ...]
func metadataItemForMediaContentRating(_ rating: String) -> AVMetadataItem {
    let metadataItem = AVMutableMetadataItem()
    metadataItem.value = value as NSString
    metadataItem.extendedLanguageTag = "und"
    metadataItem.identifier = .iTunesMetadataContentRating
    return metadataItem
}

let playerItem = AVPlayerItem(url: url)

// World-wide content ratings may be used; the user’s rating region will be preferred.
playerItem.externalMetadata = [metadataItemForMediaContentRating("PG-13"), ...]
// Providing media content rating metadata

func metadataItemForMediaContentRating(_ rating: String) -> AVMetadataItem {
    let metadataItem = AVMutableMetadataItem()
    metadataItem.value = value as NSString
    metadataItem.extendedLanguageTag = "und"
    metadataItem.identifier = .iTunesMetadataContentRating
    return metadataItem
}

let playerItem = AVPlayerItem(url: url)
// World-wide content ratings may be used; the user’s rating region will be preferred.
playerItem.externalMetadata = [metadataItemForMediaContentRating("PG-13"), ...]
// Using requestPlaybackRestrictionsAuthorization

playerItem?.requestPlaybackRestrictionsAuthorization({ (success, error) in
    if success {
        // If the request succeeded, hand over the player and start playback.
        // (You may wish to wait to do this until the player status is .ReadyToPlay)
        playerViewController?.player = player
        player?.rate = 1.0
    } else {
        // If the request failed, immediately dismiss our view controller.
    }
})
Demo
Best Practices on tvOS

Controls revealed by swipe-up? Use a custom overlay instead
Best Practices on tvOS

Controls revealed by swipe-up? Use a custom overlay instead

Use the delegate dismissal notifications rather than a `UITapGestureRecognizer`
Best Practices on tvOS

Controls revealed by swipe-up? Use a custom overlay instead

Use the delegate dismissal notifications rather than a UITapGestureRecognizer

Avoid toggling showsPlaybackControls during presentation
Best Practices on tvOS

Controls revealed by swipe-up? Use a custom overlay instead.

Use the delegate dismissal notifications rather than a `UITapGestureRecognizer`.

Avoid toggling `showsPlaybackControls` during presentation.

Provide media content ratings.
Best Practices on tvOS

Controls revealed by swipe-up? Use a custom overlay instead

Use the delegate dismissal notifications rather than a `UITapGestureRecognizer`

Avoid toggling `showsPlaybackControls` during presentation

Provide media content ratings

Test with parental content restrictions enabled
Summary

Adopt AVPlayerViewController
Summary

Adopt AVPlayerViewController

Use AVPlayerViewControllerDelegate notifications
Summary

Adopt AVPlayerViewController

Use AVPlayerViewControllerDelegate notifications

Observe the player item status and handle errors
Summary

Adopt AVPlayerViewController

Use AVPlayerViewControllerDelegate notifications

Observe the player item status and handle errors

Use externalMetadata on iOS and tvOS
Summary

Adopt AVPlayerViewController

Use AVPlayerViewControllerDelegate notifications

Observe the player item status and handle errors

Use externalMetadata on iOS and tvOS

Migrate swipe-up actions to custom overlays
Summary

Adopt AVPlayerViewController

Use AVPlayerViewControllerDelegate notifications

Observe the player item status and handle errors

Use externalMetadata on iOS and tvOS

Migrate swipe-up actions to custom overlays

Support parental content restrictions
More Information

developer.apple.com/wwdc19/503

<table>
<thead>
<tr>
<th>Lab</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>AVKit Lab</td>
<td>Thursday, 2:00</td>
</tr>
<tr>
<td>AVFoundation Lab</td>
<td>Thursday, 4:00</td>
</tr>
</tbody>
</table>
Apple WWDC19