Live Screen Broadcast with ReplayKit

Alexander Subbotin, ReplayKit Engineering
ReplayKit

Capture
- Screen visuals
- App audio
- Microphone input

Record and share

Broadcast live
ReplayKit

HD quality capture
Low latency
Low performance impact
Minimal power usage
Privacy safeguards
Agenda

- Live broadcast overview
- System broadcast picker
- Developing broadcast extensions
- Protecting content
Live Broadcast Overview
Broadcast live to third party broadcast services

Stream audio and visuals directly from device

Provide commentary with microphone and camera (iOS)

Content is secure and only accessible to the broadcast service
Live Broadcast
Usage examples

Stream gameplay to Mobcrush or YouTube
Mirror screen on a WebEx call
Work with customer support via TeamViewerQS
Stream a drawing app to Facebook
Live Broadcast
ReplayKit versus ReplayKit 2

Your App ➔ Start/stop ➔ ReplayKit API ➔ A/V samples ➔ Broadcaster ➔ “In-App Broadcast”
Live Broadcast

ReplayKit versus ReplayKit 2

Your App → ReplayKit API → Broadcaster
Start/stop ➔ A/V samples

“In-App Broadcast”

Control Center → ReplayKit 2 Session → Broadcaster
Start/stop ➔ A/V samples

“iOS System Broadcast”
In-App Broadcast

Your app or game

- Provides the content—visuals and audio
- Starts and stops the broadcast

Broadcaster app

- Provides sign-in and upload extensions
- Streams content to their network
iOS System Broadcast

Broadcasts all onscreen activity (and sounds)
Start and stop from Control Center
Systemwide, continuous session
  • Home screen
  • Moving app to app
Built-in to iOS 11 and above
Let me earn your favor, sweet queen! Send me off to slay a rare beast, and bring you a trophy!

Albert the Swift

Ijeoma
5 years
1069
Everything on your screen, including notifications, will be recorded. Enable Do Not Disturb to prevent unexpected notifications.
Let me earn your favor, sweet queen! Send me off to slay a rare beast, and bring you a trophy!

Albert the Swift

Ijeoma
5 years
1069
iOS System Broadcast

Broadcast application

Let me earn your favor, sweet queen! Send me off to play a rare beast and bring you a trophy!

Nathan
Can't wait to play, looks awesome 👍!!!!

Violet
😃😄😆😅😂!!!

Zach
Always love your stuff, can't wait to see what kind of sweet tricks you have for us

Carlos
🙂🙃😉!!!

Paula
So much fun!! ♥♥♥♥

Add comment
iOS System Broadcast

Broadcast application

Amanda Archer Extraordinaire

Nathan Can't wait to play, looks awesome 😍👍!!

Violet 😃😄😆😅😂!!!

Zach Always love your stuff, can't wait to see what kind of sweet tricks you have for us 😃🙃😉!!!

Carlos 😊🙃😉!!!

Paula So much fun! 😍♥️♥️♥️♥️

Subscribe

Live 92,364 Viewers
System Broadcast Picker
ReplayKit API

Your App

ReplayKit API

Broadcaster

“In-App Broadcast”

Control Center

ReplayKit 2 Session

Broadcaster

“iOS System Broadcast”
In-App Broadcast

iOS System Broadcast

Your App

ReplayKit API

Broadcaster

Control Center

ReplayKit 2 Session

Broadcaster

A/V samples

Start/stop

A/V Samples

NEW
Broadcast Picker

Your app can initiate an iOS system broadcast session

Simple one-button UI

No compromises for privacy

Secure architecture

New in iOS 12
Fox 2
Everything on your screen, including notifications, will be recorded. Enable Do Not Disturb to prevent unexpected notifications.
class RPSsystemBroadcastPickerView: UIView {
    open var preferredExtension: String
}
import ReplayKit.broadcast

class ViewController: UIViewController {
    var broadcastPicker: RPSystemBroadcastPickerView?

    override func viewDidLoad() {
        super.viewDidLoad()
        broadcastPicker = RPSystemBroadcastPickerView(frame: kPickerFrame)
        view.addSubview(broadcastPicker)
    }
}
Screen Broadcast

TeamViewer

Start Broadcast

Microphone
Audio
Off
preferredExtension

Pairs broadcast picker to a particular extension

Assign bundle identifier of your extension

Initialize before view is presented
import ReplayKit.broadcast

class ViewController: UIViewController {
    var broadcastPicker: RPSystemBroadcastPickerView?

    override func viewDidLoad() {
        super.viewDidLoad()
        broadcastPicker = RPSystemBroadcastPickerView(frame: kPickerFrame)
        broadcastPicker.preferredExtension = "com.your-app.broadcast.extension"

        view.addSubview(broadcastPicker)
    }
}
Brings up system broadcast picker
Allows to configure broadcast picker
Doesn’t own any state of the session
Developing Broadcast Extensions
For ReplayKit 2
Control Center ➔ Start/stop ➔ ReplayKit 2 Session ➔ A/V samples ➔ Broadcaster ➔ “iOS System Broadcast”
Broadcast App and Extension

Broadcast Application
• Account sign-in, broadcast title

Broadcast Upload Extension
• Encode samples, upload to service

Diagram:
- Broadcast application
- Upload extension
- Media stream
- Audio and video samples
- Login credentials, etc
Broadcast Upload Extension

- Receives audio and video samples
- Encodes and uploads video stream
- Handles device orientation changes
- Annotates broadcast with app information

[Diagram showing audio and video samples flowing through an upload extension to a media stream]

V A A

Upload extension

Audio and video samples

Media stream
Broadcast Extension Template
class SampleHandler: RPBroadcastSampleHandler {
    // User has requested to start the broadcast
    override func broadcastStarted(withSetupInfo setupInfo: [String : NSObject]?)
    // User has requested to finish the broadcast
    override func broadcastFinished()
    // Handle the sample buffer here
    override func processSampleBuffer(_ sampleBuffer: CMSampleBuffer,
                                        with sampleBufferType: RPSampleBufferType)
    // Use details of application to annotate the broadcast
    override func broadcastAnnotated(withApplicationInfo info: [String : NSObject])
}
Handling Sign-In and Broadcast Setup

Broadcast application

Setup

- Initialized
- Started
- Process Sample
- Stopped
Initialization

Application

Setup

Initialized

Started

ProcessSample

Stopped
// Override init to read login credentials from shared keychain

class SampleHandler : RPBroadcastSampleHandler {
    override func init() {
        super.init()
        session = BroadcastSession.instance
        var credentials = KeychainAccess.getLoginCredentials()
        session.authentificate(credentials)
    }
}
Handling `broadcastStarted`
// Override broadcastStarted to prepare to receive media samples
override func broadcastStarted(withSetupInfo setupInfo: [String : NSObject]?) {
    // Verify user is logged in
    if (session.userLoggedIn()) {
        session.createMediaEngine()
    }
}
Processing Media Samples

Application

Setup

Initialized

Started

Extension

Process Sample

Stopped
Broadcast Upload Extension

processSampleBuffer

- Video (screen)
- Audio (app)
- Audio (microphone)

Upload extension

Media stream

Video Toolbox
// Both audio and video samples are handled by processSampleBuffer routine

override func processSampleBuffer(_ sampleBuffer: CMSampleBuffer,
                                       with sampleBufferType: RPSampleBufferType) {

    switch sampleBufferType {
    case RPSampleBufferType.video:
        var imageBuffer:CVImageBuffer = CMSampleBufferGetImageBuffer(sampleBuffer)!
        var pts = CMSampleBufferGetPresentationTimeStamp(sampleBuffer) as CMTime
        VTCompressionSessionEncodeFrame(session, imageBuffer, pts,
                                          kCMTimeInvalid, nil, nil, nil)
        break

    case RPSampleBufferType.audioApp:
        // Handle audio sample buffer for app audio
        break

    case RPSampleBufferType.audioMic:
        // Handle audio sample buffer for mic audio
        break
    }
}
Handling Application Information

Application:
- Setup
- Initialized
- Started

Extension:
- Process sample
- Stopped
Handling Application Information

Application

- Setup
- Initialized
- Started
- Process sample
- Stopped

Extension

- App information
Broadcast Upload Extension

broadcastAnnotatedWithApplicationInfo

ReplayKit -> Upload extension

Home Screen
Angry Birds 2
Second Application
Third Application
Fourth Application

Elapsed time

Media stream
override func broadcastAnnotated(withApplicationInfo applicationInfo: [AnyHashable : Any]) {
    var bundleIdentifier = applicationInfo[RPApplicationInfoBundleIdentifierKey]
    if (bundleIdentifier != nil) {
        session.addMetadataWithApplicationInfo(bundleIdentifier)
    }
}
Handling broadcastFinished

Application

Setup

Initialized

Started

Extension

Process sample

App information

Stopped
PLAY

Livestream your gameplay everywhere at once.

Sign up

Sign In

By signing up, you agree to our Terms & Privacy Policy.
override func broadcastStarted(withSetupInfo setupInfo: [String : NSObject]?) {
    // Verify user is logged in and there's network connectivity
    if (session.userLoggedIn()) {
        session.createMediaEngine()
    } else {
        let userInfo = [NSLocalizedFailureReasonErrorKey : "Not Logged In"]
        let error = NSError(domain: "RPBroadcastErrorDomain", code: 401, userInfo: userInfo)
        finishBroadcastWithError(error)
    }
}
Implement sign-in and setup UI in your application

Finish the broadcast and send user to the app if something is missing

Encode and upload video stream to your service

Use information about app on the screen to help users find your broadcast
Protecting Content
Protecting Content

UIScreen.isCaptured

Prevent capturing of audio and video content of your app

Stop media playback or displaying sensitive content

• Check value of UIScreen.captured
• Register for UIScreenCapturedDidChangeNotification
• Check UIScreen.screens.count to allow screen mirroring
import UIKit

class func handleScreenCapturedChange() {
    let isScreenMirroring = UIScreen.screens.count > 1
    if (UIScreen.isCaptured && !isScreenMirroring) {
        // stop audio playback and remove sensitive content from the screen
    }
}

// Protecting content of your application from being captured
Summary

Live screen broadcast
System broadcast picker
Developing broadcast extensions
Protecting content
More Information


ReplayKit Lab

Technology Lab 5

Tuesday 3:00PM