Running **Radio** in the Background

This is a short manual how to run Radio in the background on standalone, iOS or Android.

crosstales LLC creates assets to help you realize great solutions in **Unity**:

Visit us at the Unity AssetStore

Disclaimer:

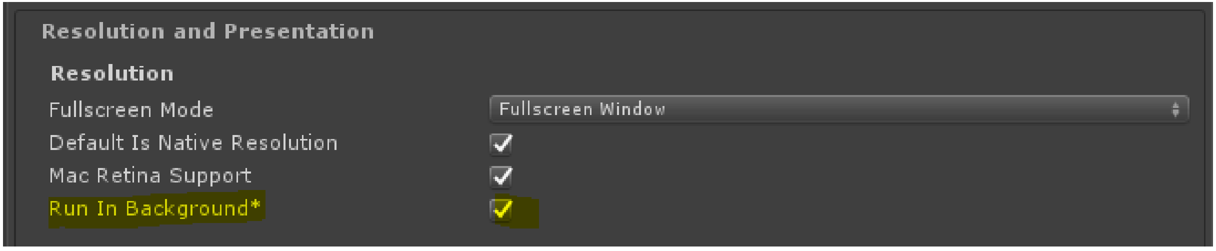
Unity is **not designed** to work in **background** on **mobile** platforms. The following pages represent some kind of a hack to achieve background play and it is entirely possible that it would not work.

Note:

In this tutorial we used
- **Unity 2018.4.14**
- **Xcode 10.1** (tested with **iOS 14.4**)
- **Android Studio 4.1.2** (tested with **Android 11**)

Standalone

1. Go to the **PlayerSettings**
   File -> Build Settings -> PlayerSettings
2. Select tab **Other Settings**
3. Enable "Run In Background":



Resolution and Presentation

Resolution

Fullscreen Mode Fullscreen Window

Default Is Native Resolution ☑

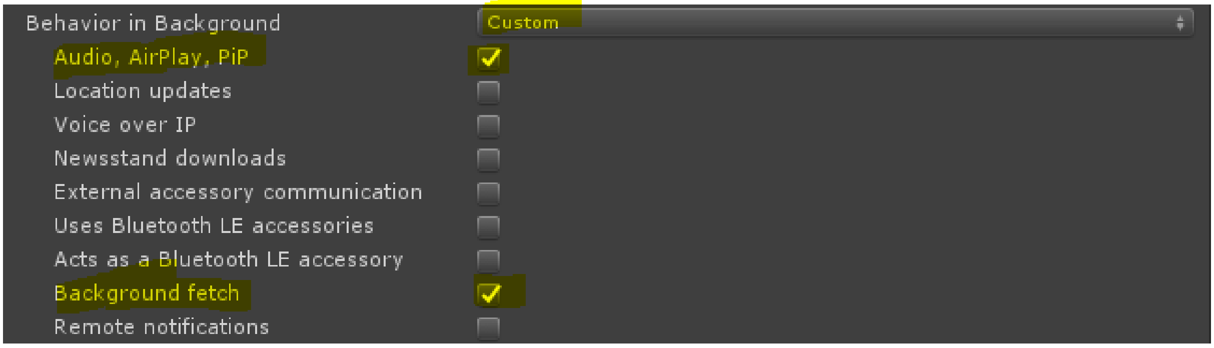
Mac Retina Support ☑

Run In Background* ☑

iOS

General steps

1. Go to the **PlayerSettings**
   File -> Build Settings -> PlayerSettings
2. Select tab **Other Settings**
3. Change the "Behaviour in Background" from ***Suspend*** to ***Custom***
4. Set the "Audio, AirPlay, PiP" and "Background fetch" active



Behavior in Background: Custom
- ☑ Audio, AirPlay, PiP
- □ Location updates
- □ Voice over IP
- □ Newsstand downloads
- □ External accessory communication
- □ Uses Bluetooth LE accessories
- □ Acts as a Bluetooth LE accessory
- ☑ Background fetch
- □ Remote notifications

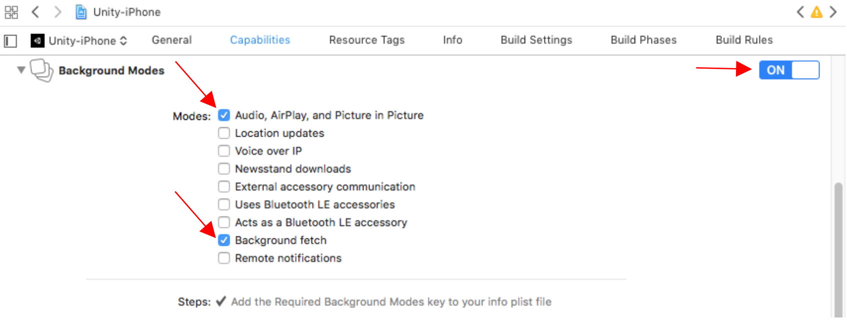
5. Build the project for **iOS**

New way (in Unity)

1. Import the package iOS_RunInBackground.unitypackage from "Assets\Plugins\crosstales\Common\Extras"
2. Build the project for **iOS**

Old way (in Xcode)

1. Build the project for **iOS**
2. Open the Xcode project
3. Click on the project (with the blue icon to the left) and open the **Capabilities** tab
4. Scroll to the **Background Modes** and check if the "Audio, AirPlay, and Picture in Picture" and "Background fetch" are **active**:

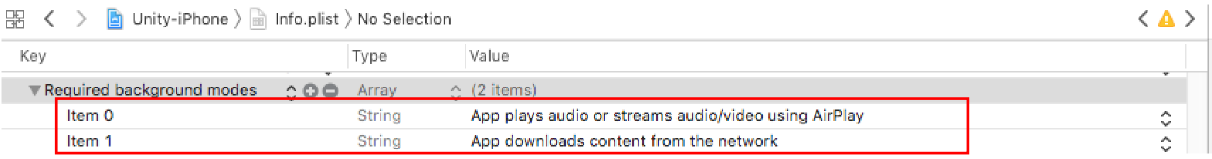


Background Modes: ON

Modes:
- ☑ Audio, AirPlay, and Picture in Picture
- □ Location updates
- □ Voice over IP
- □ Newsstand downloads
- □ External accessory communication
- □ Uses Bluetooth LE accessories
- □ Acts as a Bluetooth LE accessory
- ☑ Background fetch
- □ Remote notifications

Steps: ✓ Add the Required Background Modes key to your info plist file

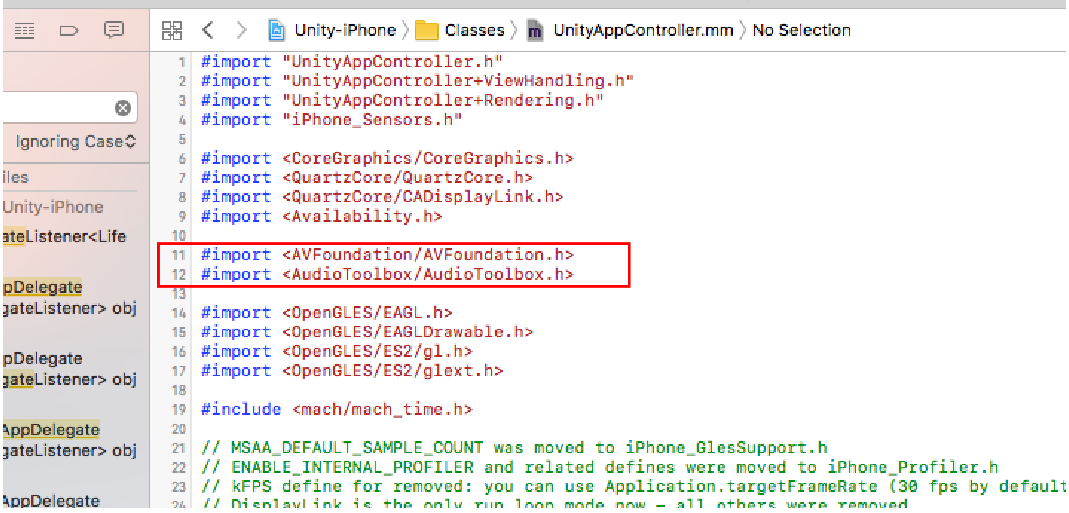
5. Open the ***info.plist*** file and check the following values:



Key	Type	Value
▼ Required background modes	Array	(2 items)
Item 0	String	App plays audio or streams audio/video using AirPlay
Item 1	String	App downloads content from the network

6. Open the ***UnityAppController.mm*** file
7. Add the **import** lines for **AVFoundation** and **AudioToolbox**:

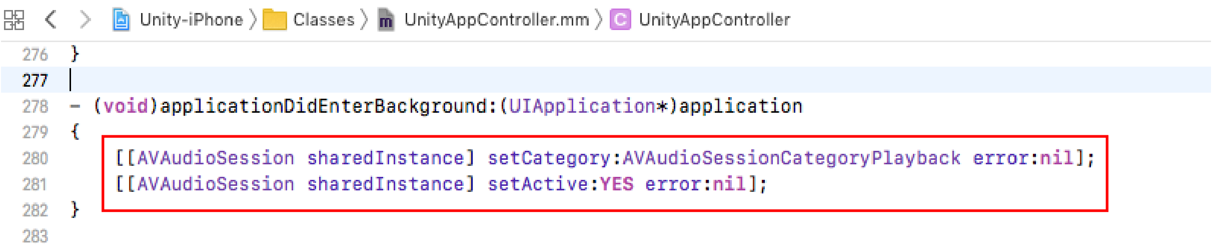
```
#import <AVFoundation/AVFoundation.h>

#import <AudioToolbox/AudioToolbox.h>
```



```
#import "UnityAppController.h"
#import "UnityAppController+ViewHandling.h"
#import "UnityAppController+Rendering.h"
#import "iPhone_Sensors.h"

#import <CoreGraphics/CoreGraphics.h>
#import <QuartzCore/QuartzCore.h>
#import <QuartzCore/CADisplayLink.h>
#import <Availability.h>

#import <AVFoundation/AVFoundation.h>
#import <AudioToolbox/AudioToolbox.h>

#import <OpenGLES/EAGL.h>
#import <OpenGLES/EAGLDrawable.h>
#import <OpenGLES/ES2/gl.h>
#import <OpenGLES/ES2/glext.h>

#include <mach/mach_time.h>

// MSAA_DEFAULT_SAMPLE_COUNT was moved to iPhone_GlesSupport.h
// ENABLE_INTERNAL_PROFILER and related defines were moved to iPhone_Profiler.h
// kFPS define for removed: you can use Application.targetFrameRate (30 fps by default
// DisplayLink is the only run loop mode now - all others were removed
```

8. Next, search in the same file for "**applicationDidEnterBackground**" and add these two lines:

```
[[AVAudioSession sharedInstance] setCategory:AVAudioSessionCategoryPlayback error:nil];

[[AVAudioSession sharedInstance] setActive:YES error:nil];
```


Unity-iPhone 〉 Classes 〉 UnityAppController.mm 〉 UnityAppController

```
}

- (void)applicationDidEnterBackground:(UIApplication*)application
{
    [[AVAudioSession sharedInstance] setCategory:AVAudioSessionCategoryPlayback error:nil];
    [[AVAudioSession sharedInstance] setActive:YES error:nil];
}
```

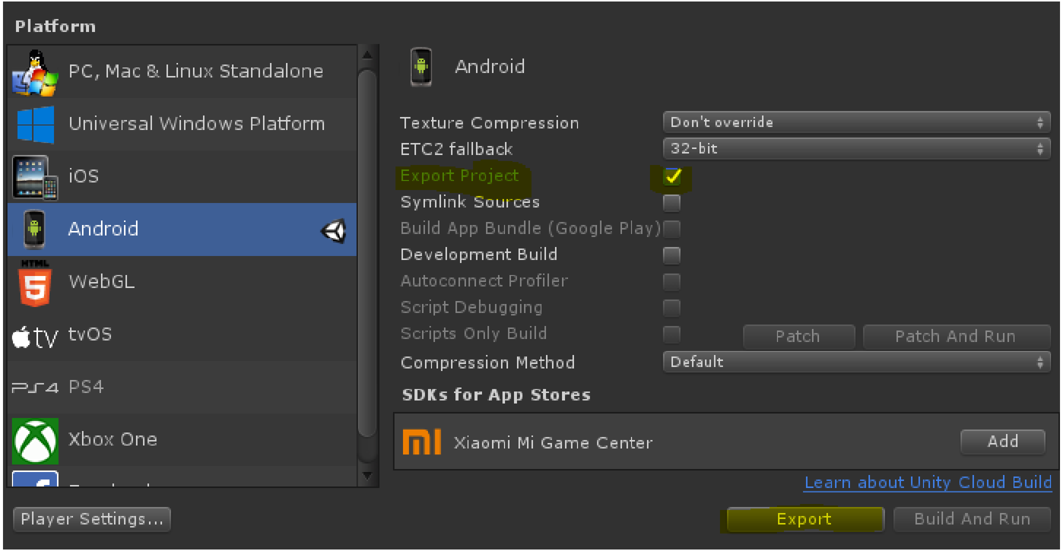
Android

New way (in Unity)

1. Import the package **Android_RunInBackground.unitypackage** from “Assets\Plugins\crosstales\Common\Extras”
2. Optional: configure the **AndroidManifest.xml** under “Assets\Plugins\Android”
3. Optional: modify the **CTUnityPlayerActivity.java** under “Assets\Plugins\Android”
4. Build the project for **Android**

Old way
Unity Settings

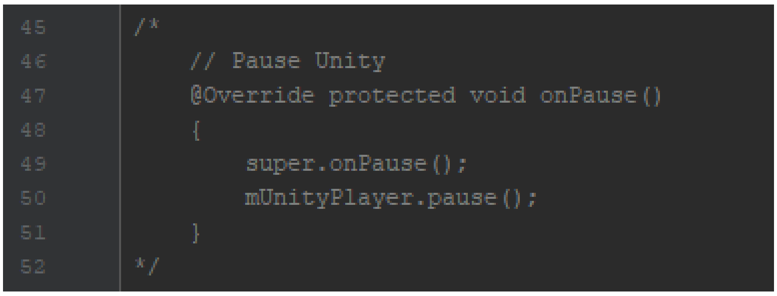
1. Open the **Build Settings -> File -> Build Settings**
2. Set *Export Project* to **active**
3. Click **Export** and choose a folder



Texture Compression: Don't override
ETC2 fallback: 32-bit
Export Project ☑
Symlink Sources □
Build App Bundle (Google Play) □
Development Build □
Autoconnect Profiler □
Script Debugging □
Scripts Only Build □
Compression Method: Default

Android Studio Settings

4. Open **Android Studio**
5. Open the project folder
6. Open the ***UnityPlayerActivity.java*** file
7. Search for “onPause”
8. Comment the method:



```
/*
    // Pause Unity
    @Override protected void onPause()
    {
        super.onPause();
        mUnityPlayer.pause();
    }
*/
```