intel Sports

INTEL® IMMERSIVE EXPERIENCE SDK FOR MOBILE V1.9

Quick Start Guide

TABLE OF CONTENTS

INTRODUCTION

The Intel® Immersive Experience SDK for Mobile allows you to seamlessly and easily integrate cutting-edge content into your application. Your audience will enjoy an amazing replays and highlights experience that brings together game statistics and data with the most vantage points for the viewing pleasure. Application users may choose from multiple cameras and points of view, and even from a players' perspective.

This guide provides step-by-step directions for creating a fully functional VR video player embedded into your application, using either an Android or iOS wrapper. Render Intel® Sports content in you app - everything from 2D (monoscopic) panoramic videos, 3D (stereoscopic) VR videos, as well as 360⁰ videos and images.

REQUIREMENTS FOR iOS

Xcode 10 and iOS 10 or above - press here to acquire

Minimum Intel® supported iOS™ version - iOS 10.13.6

Intel® Immersive Experience SDK framework kit archive

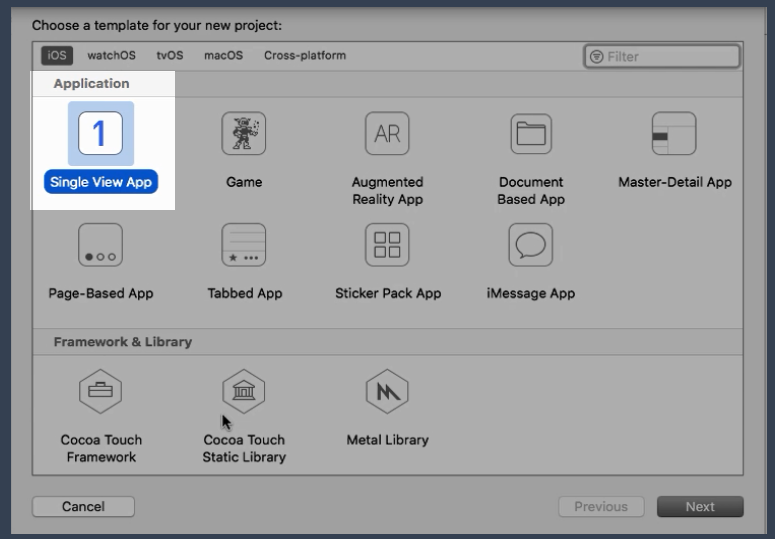
Overall procedure time: ~ 6-10 min

* For the full guidance read User Manual_iOS.doc

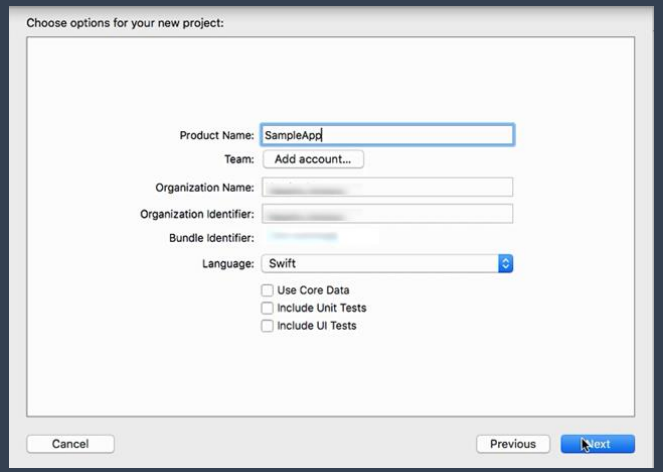
STEP 1 : SETUP PROJECT

iOS

1. Create a new Xcode project
2. Select a Single View App template for the project



3. In the new project options window:
   i. Type Product Name for your project
   ii. Select Language as Objective-C or Swift

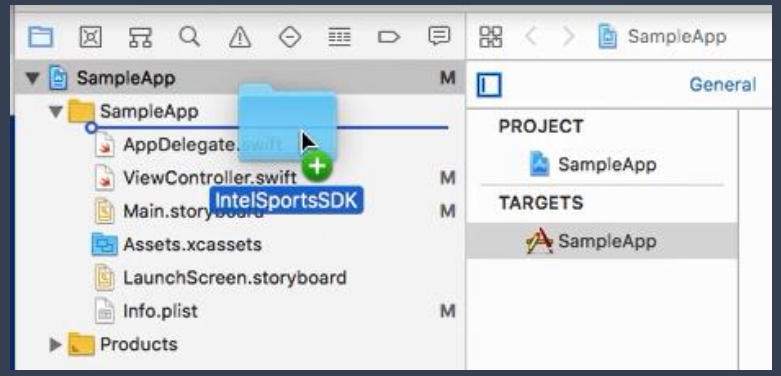


4. Verify Devices set to Universal under Deployment Info

STEP 2 : ADD FRAMEWORK

iOS

1 Drag and drop the IntelSportsSDK directory (from Intel® Sports framework kit) to the Project Navigator



2 In adding file options window:

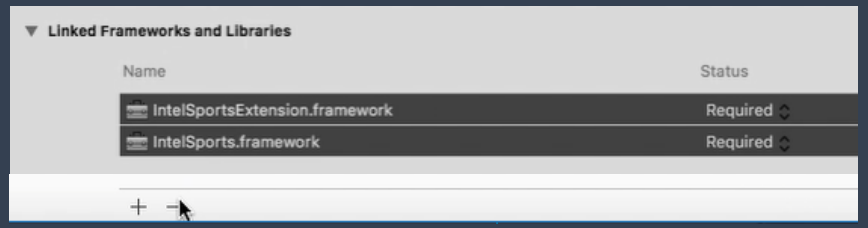
i. Add checkmark to Destination: Copy items if needed

ii. Mark Added folders: Create groups

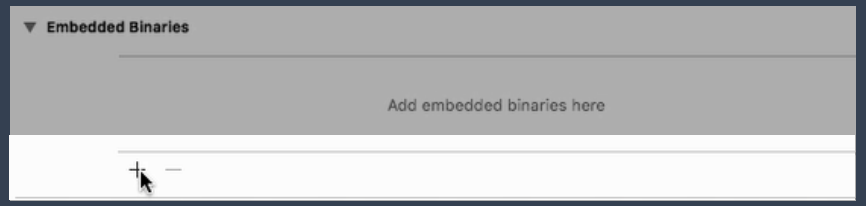
STEP 2 : ADD FRAMEWORK

iOS

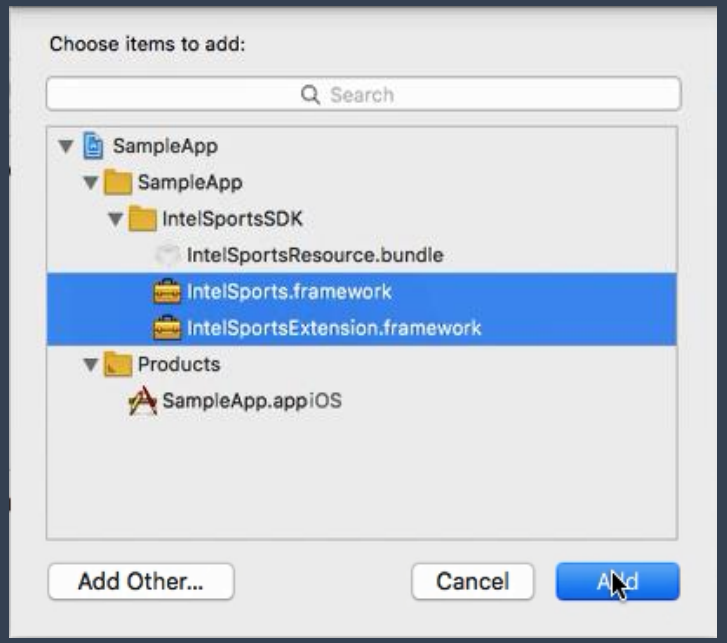
3 Select IntelSports.framework & IntelSportsExtension.framework under General > Linked frameworks and Libraries, and press (-) icon to remove



4 Press (+) icon under Embedded Binaries to add items



5 Choose IntelSports.framework & IntelSportsExtension.framework to add to Embedded Binaries



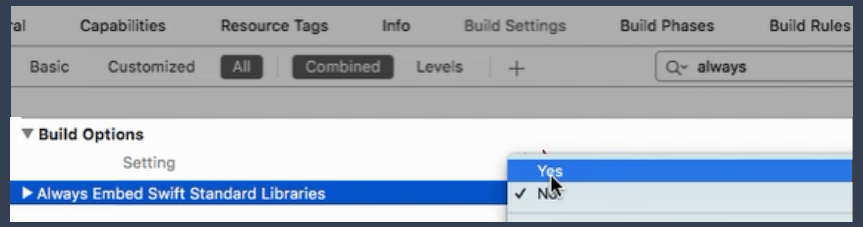
STEP 2 : ADD FRAMEWORK

iOS

6 For developers using Objective C set

Always Embed Swift Standard Libraries to Yes under

Build Settings > Build Options



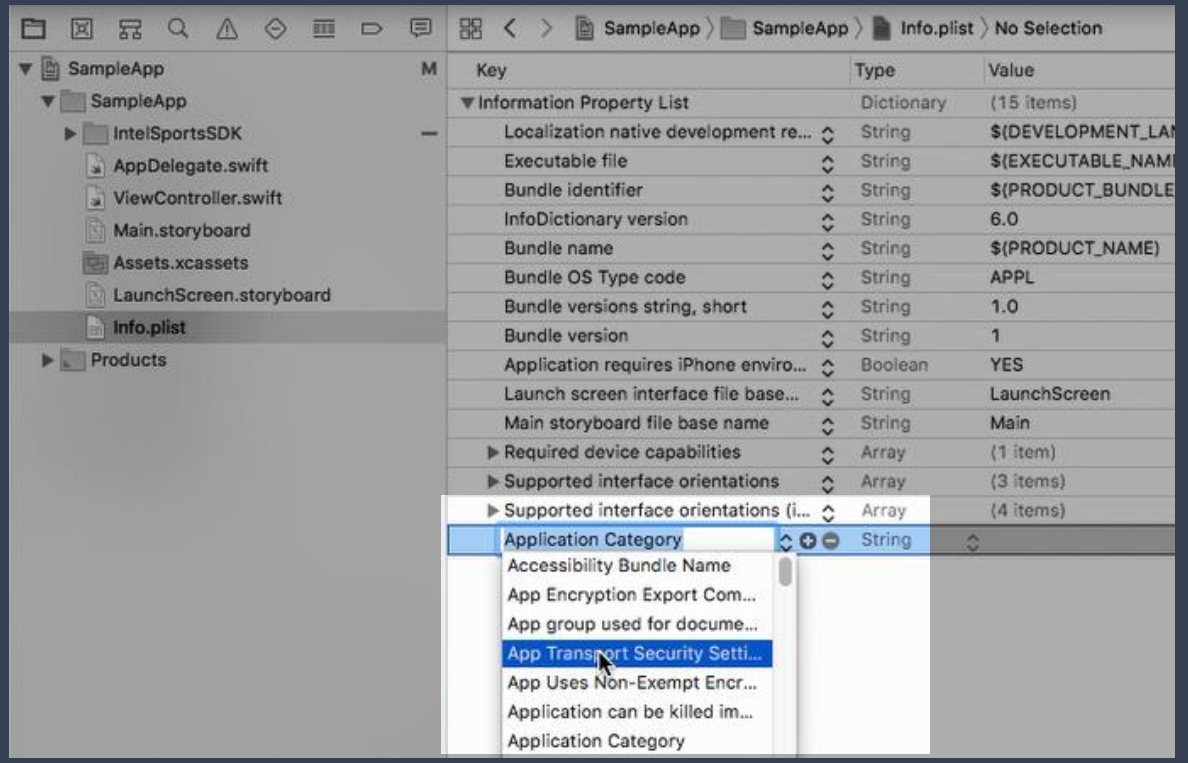
STEP 3 : ADD TRANSPORT SECURITY

iOS

To support iOS 10 and above add App Transport Security key.

1 In Info.plist > Supported interface orientations, press (+) to add App Transport Security Settings

> Note:
>
> This step is required only if the video source URL is not secured (https).

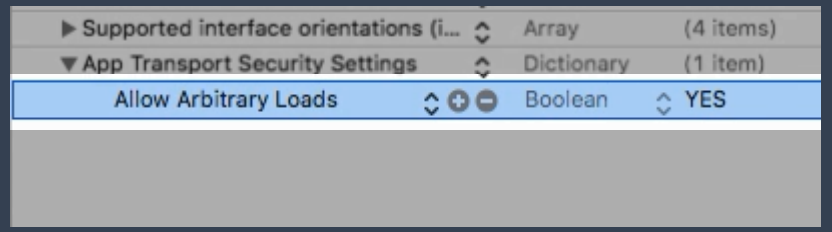


STEP 3 : ADD TRANSPORT SECURITY

iOS

2 Press ▶ icon next to App Transport Security Settings

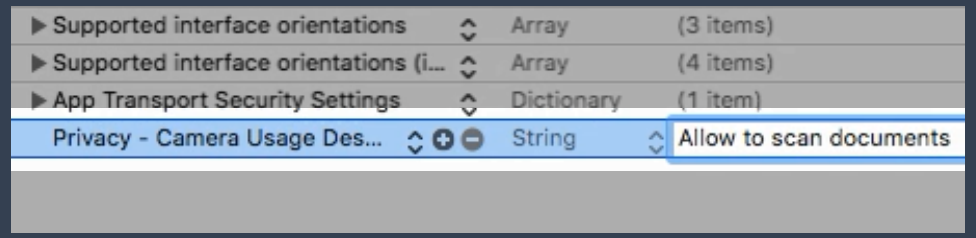
3 Press (+) icon to add application category, and choose Allow Arbitrary Loads (set Yes)



To support iOS 10.0 and above add Privacy - Camera Usage Description key

4 Press (+) icon to add new application category, and choose Privacy - Camera Usage Description

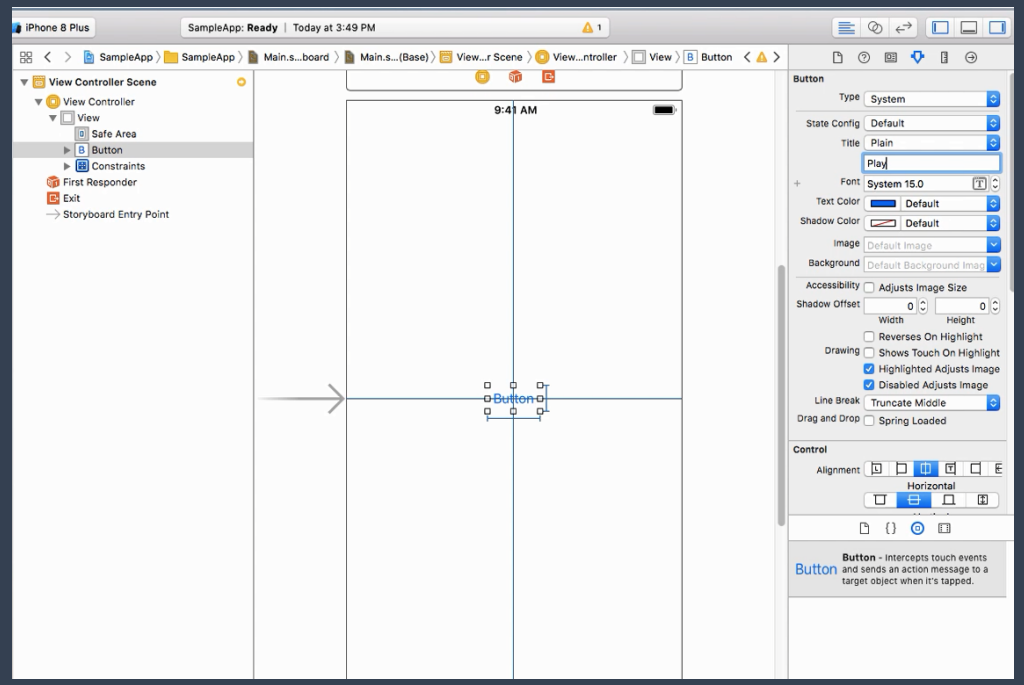
i. Type value Allow to scan documents



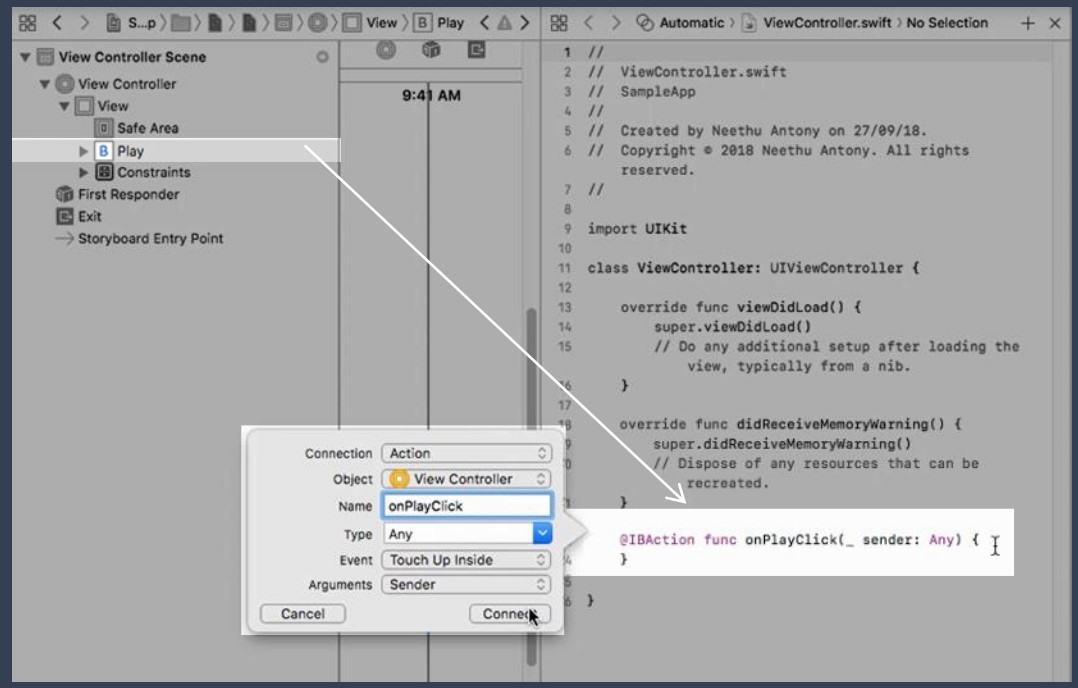
STEP 4 : CREATE THE VIDEO CONTROLLER UI

iOS

1 Add Play action Button to the Main.storyboard



2 Connect onPlayClick action for the button to the ViewController.swift



STEP 4 : CREATE THE VIDEO CONTROLLER UI

iOS

3 Add the following header lines to the ViewController.swift file:

```
import IntelSports
import IntelSportsExtension
```

SampleApp 〉 SampleApp 〉 ViewController.swift 〉 No Selection

```
1  //
2  //  ViewController.swift
3  //  SampleApp
4  //
5  //  Created by Neethu Antony on 27/09/18.
6  //  Copyright © 2018 Neethu Antony. All rights reserved.
7  //
8
9  import UIKit
10 import IntelSports
11 import IntelSportsExtension
12
13 class ViewController: UIViewController {
14
15     override func viewDidLoad() {
16         super.viewDidLoad()
17         // Do any additional setup after loading the view, typically from a nib.
18     }
19
20     override func didReceiveMemoryWarning() {
21         super.didReceiveMemoryWarning()
22         // Dispose of any resources that can be recreated.
23     }
24
25     @IBAction func onPlayClick(_ sender: Any) {
26
27     }
28
29 }
```

4 Add ISViewControllerdelegate interface line to the ViewController class

```
ISViewControllerDelegate
```

SampleApp 〉 SampleApp 〉 ViewController.swift 〉 ViewController

```
1  //
2  //  ViewController.swift
3  //  SampleApp
4  //
5  //  Created by Neethu Antony on 27/09/18.
6  //  Copyright © 2018 Neethu Antony. All rights reserved.
7  //
8
9  import UIKit
10 import IntelSports
11 import IntelSportsExtension
12
13 class ViewController: UIViewController, ISViewControllerDelegate {
14
15     override func viewDidLoad() {
16         super.viewDidLoad()
17         // Do any additional setup after loading the view, typically from a nib.
18     }
19
20     override func didReceiveMemoryWarning() {
21         super.didReceiveMemoryWarning()
22         // Dispose of any resources that can be recreated.
23     }
24
25     @IBAction func onPlayClick(_ sender: Any) {
26
27     }
```

STEP 4 : CREATE THE VIDEO CONTROLLER UI

iOS

5 Copy the following different CameraInfos to switch between streams.

```
let camera1 = CameraInfo()
camera1.urlMonoNormal =
"http://live.vokevr.com/2018MLB/20180423NationalsAtGiants/monoscopic/pod5/mono_master_normal.m3u8";
camera1.cameraName = "Pod 1";
camera1.videoMappingType = MappingType.cylindrical;

let camera2 = CameraInfo()
camera2.urlMonoHigh =
"http://live.vokevr.com/2018MLB/20180423NationalsAtGiants/monoscopic/pod5/mono_master_high.m3u8";
camera2.cameraName = "Pod 2";
camera2.videoMappingType = MappingType.cylindrical;

let camera3 = CameraInfo()
camera3.urlMonoHigh =
"http://live.vokevr.com/2018MLB/20180423NationalsAtGiants/monoscopic/pod1/mono_master_high.m3u8";
camera3.cameraName = "Pod 3";
camera3.videoMappingType = MappingType.cylindrical;

let camera4 = CameraInfo()
camera4.urlMonoNormal =
"http://live.vokevr.com/2018MLB/20180423NationalsAtGiants/monoscopic/pod1/mono_master_normal.m3u8";
camera4.cameraName = "Pod 4";
camera4.videoMappingType = MappingType.cylindrical;

let videoInfo = VideoInfo()
videoInfo.add(camera1)
videoInfo.add(camera2)
videoInfo.add(camera3)
videoInfo.add(camera4)
```

6 Paste the CameraInfos to the onPlayClick

SampleApp 〉 SampleApp 〉 ViewController.swift 〉 onPlayClick(_:)

```
1  //
2  //  ViewController.swift
3  //  SampleApp
4  //
5  //  Created by Neethu Antony on 27/09/18.
6  //  Copyright © 2018 Neethu Antony. All rights reserved.
7  //
8
9  import UIKit
10 import IntelSports
11 import IntelSportsExtension
12
13 class ViewController: UIViewController, ISViewControllerDelegate {
14
15     override func viewDidLoad() {
16         super.viewDidLoad()
17         // Do any additional setup after loading the view, typically from a nib.
18     }
19
20     override func didReceiveMemoryWarning() {
21         super.didReceiveMemoryWarning()
22         // Dispose of any resources that can be recreated.
23     }
24
25     @IBAction func onPlayClick(_ sender: Any) {
26
27     }
28
29 }
30
31
```

EXTRA : CAMERA SWITCHING

iOS

* For the extra Camera Switching functionality, which enables end user to switch between the video streams, load up more CameraInfo URLs to the onPlayClick.

Note:

In panoramic viewing mode, the SDK will render the top image of a stereoscopic top/bottom video source, if a monoscopic URL is not passed to the CameraInfo class as a separate video source.

MonoHigh (Low Home camera):
http://live.vokevr.com/2018MLB/20180423NationalsAtGiants/monoscopic/pod2/mono_master_high.m3u8

MonoNormal (Low Home camera):
http://live.vokevr.com/2018MLB/20180423NationalsAtGiants/monoscopic/pod2/mono_master_normal.m3u8

MonoHigh (First Base camera):
http://live.vokevr.com/2018MLB/20180423NationalsAtGiants/monoscopic/pod1/mono_master_high.m3u8

MonoNormal (First Base camera):
http://live.vokevr.com/2018MLB/20180423NationalsAtGiants/monoscopic/pod1/mono_master_normal.m3u8

MonoHigh (Third Base camera):
http://live.vokevr.com/2018MLB/20180423NationalsAtGiants/monoscopic/pod3/mono_master_high.m3u8

MonoNormal (Third Base camera):
http://live.vokevr.com/2018MLB/20180423NationalsAtGiants/monoscopic/pod3/mono_master_normal.m3u8

MonoHigh (Dug Out camera):
http://live.vokevr.com/2018MLB/20180423NationalsAtGiants/monoscopic/pod4/mono_master_high.m3u8

MonoNormal (Dug Out camera):
http://live.vokevr.com/2018MLB/20180423NationalsAtGiants/monoscopic/pod4/mono_master_normal.m3u8

STEP 4 : CREATE THE VIDEO CONTROLLER UI

iOS

7 Copy the following script to present the ISViewController (container SDK) and paste it under the onPlayClick

```
let vc = ISViewController(nibName: "IntelSportsResource.bundle/ISViewController", bundle: nil)

 vc.videoInfo = videoInfo
 vc.delegate = self

 self.present(vc, animated: true, completion: nil)
```

SampleApp 〉 SampleApp 〉 ViewController.swift 〉 onPlayClick(_:)

```
class ViewController: UIViewController, ISViewControllerDelegate {


    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view, typically from a nib.
    }


    override func didReceiveMemoryWarning() {
        super.didReceiveMemoryWarning()
        // Dispose of any resources that can be recreated.
    }



    @IBAction func onPlayClick(_ sender: Any)
    {
        let camera1 = CameraInfo()
        camera1.urlMonoNormal = "http://live.vokevr.com/2018MLB/20180423NationalsAtGiants/monoscopic/pod5/mono_master_normal.m3u8";
        camera1.cameraName = "Pod 1";
        camera1.videoMappingType = MappingType.cylindrical;
        let camera2 = CameraInfo()
        camera2.urlMonoHigh = "http://live.vokevr.com/2018MLB/20180423NationalsAtGiants/monoscopic/pod5/mono_master_high.m3u8";
        camera2.cameraName = "Pod 2";
        camera2.videoMappingType = MappingType.cylindrical;
        let camera3 = CameraInfo()
        camera3.urlMonoHigh = "http://live.vokevr.com/2018MLB/20180423NationalsAtGiants/monoscopic/pod1/mono_master_high.m3u8";
        camera3.cameraName = "Pod 3";
        camera3.videoMappingType = MappingType.cylindrical;
        let camera4 = CameraInfo()
        camera4.urlMonoNormal = "http://live.vokevr.com/2018MLB/20180423NationalsAtGiants/monoscopic/pod1/mono_master_normal.m3u8";
        camera4.cameraName = "Pod 4";
        camera4.videoMappingType = MappingType.cylindrical;
        let videoInfo = VideoInfo()
        videoInfo.add(camera1)
        videoInfo.add(camera2)
        videoInfo.add(camera3)
        videoInfo.add(camera4)

        let vc = ISViewController(nibName:"IntelSportsResource.bundle/ISViewController", bundle: nil)
        vc.videoInfo = videoInfo
        vc.delegate=self
        self.present(vc, animated: true,completion:nil)

    }

}
```

STEP 5 : CREATE INTERFACE FOR THE MOMENTS CONTENT

iOS

1 Open the ViewController.swift file and add following header files:

```
import IntelSports
import IntelSportsExtension
```

2 Set the URLs and player infos as below:

```
let player = Player()
player.povImageUrl = "https://s3-us-west-2.amazonaws.com/trp-cms-raw-content/mock_data/500_catch_360_POV_mono.jpg"
player.profileImageUrl = "http://static.nfl.com/static/content/public/static/img/fantasy/transparent/200x200/BRA371156.png"
player.name = "Tom Brady"
player.position = "Quarterback"
moment.contextClipUrl = "https://s3-us-west-2.amazonaws.com/mobile-sdk-test-content/nfl/true_view/ISTV_WK5_IND_NE_17_CC_360_230318_360/AVC/mono_master_high.m3u8"


let player2 = Player()
player2.profileImageUrl = "http://static.nfl.com/static/content/public/static/img/fantasy/transparent/200x200/GOR190310.png"
player2.povImageUrl = "https://s3-us-west-2.amazonaws.com/trp-cms-raw-content/mock_data/500_catch_360_POV_mono.jpg"
player2.name = "Josh Gordon"
player2.position = "Wide Receiver"

let moment = Moment()
moment.contextWauwUrl = "https://s3-us-west-2.amazonaws.com/trp-cms-raw-content/mock_data/500_catch_wauw_mono.mp4"
moment.addPlayer(player: player)
moment.addPlayer(player: player2)
```

STEP 5 : CREATE INTERFACE FOR THE MOMENTS CONTENT

iOS

3 Present the ISMomentViewController (container SDK) whenever you need as below:

```
let vc = ISMomentViewController
(nibName: "IntelSportsResource.bundle/ISMomentViewController", bundle: nil)
vc.momentlist = [moment]
self.present(vc, animated: true, completion: nil)
```

Note:

In most cases, developers can download JSON response from Intel's CMS server which can be parsed directly into the instance of Moment class.

For example:

```
NSDictionary *jsonResponseFromCMS = … //NSDictionary from downloaded JSON response
Moment *moment = [Moment createInstanceWithResDict:jsonResponseFromCMS];
```

If there's an error, Moment will be nil. Otherwise, an instance of Moment will be created. Such instance would contain all properties including instances of Player.

STEP 6 : ADDING A NATIVE CONTAINER FOR LISTING INTERACTIVE REPLAY LIST

iOS

1 Open the ViewController.swift file and add following header files:

```
import IntelSports
import IntelSportsExtension
```

2 Set the CMS URL, CMS credential, CDN credentials and Present ISMomentContainerController for listing the Interactive Replay List as below:

```
let vc = ISMomentContainerController(nibName: "IntelSportsResource.bundle/ISMomentContainerController", bundle:
nil)
vc.setCMSUrl(cmsUrl: "<<CMS URL>>")
vc.setCMSCredentials(cmsUser: "<<UserName>>", cmsPassword: "<<Password>>")
vc.setCDNCredentials(cdnClientId: "<<Client ID>>", cdnSecret: "<<Password>>", cdnDuration: 300000)

self.navigationController?.pushViewController(vc, animated: true)
```

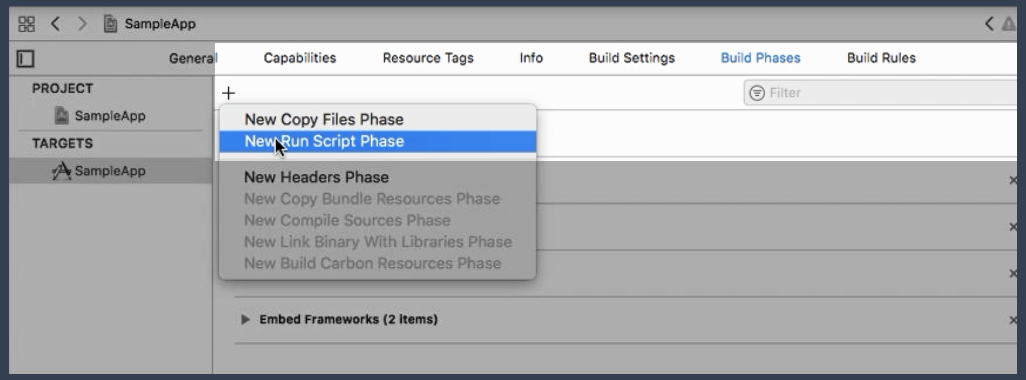
Note: The required parameters to support these implementations will be provided by Intel using key management access control methods. Please contact your Intel developer relations representative.

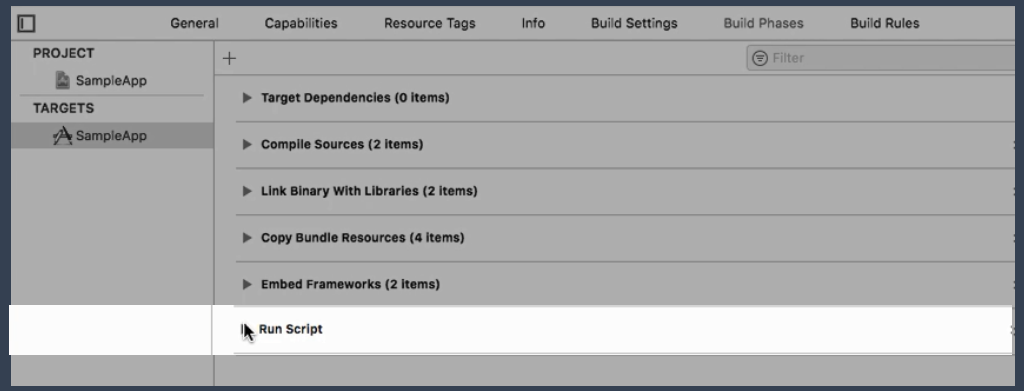
Note: The recommended access duration in the code snippet is set to 5 minutes (300 x 1000ms). This parameter should be set per your implementation requirements.

STEP 7 : STRIPPING OUT UNWANTED ARCHITECTURES

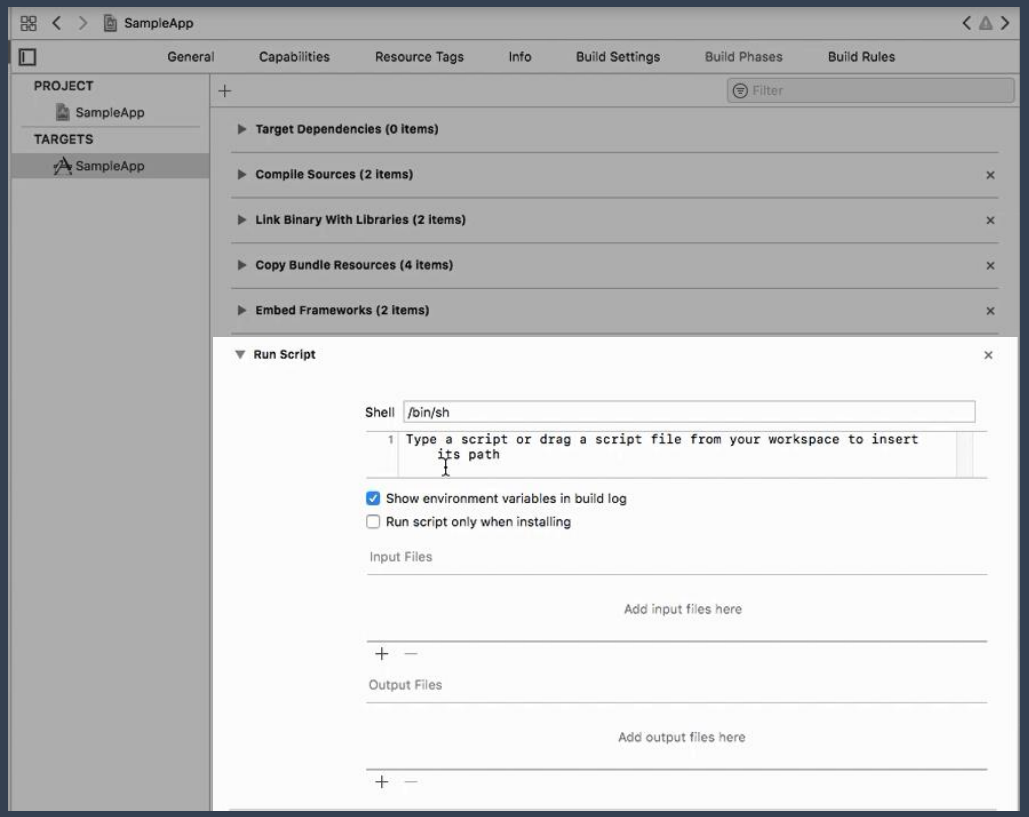
iOS

1 Press (+) icon to add New Run Script Phase under Build Phases





2 The new Run Script line is displayed, press ▶ icon to expose:



STEP 7 : STRIPPING OUT UNWANTED ARCHITECTURES

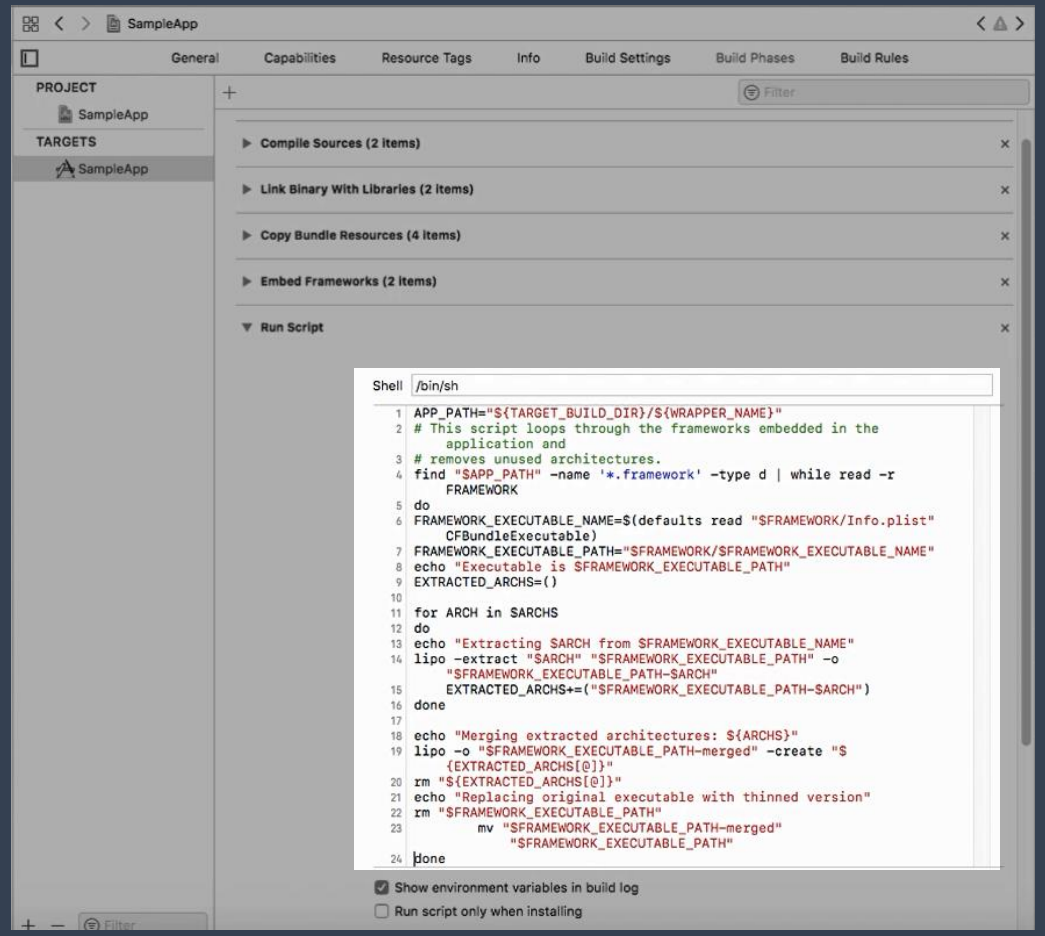
iOS

3 Copy the following script to remove unused architectures and paste in the new Run Script:

```
APP_PATH="${TARGET_BUILD_DIR}/${WRAPPER_NAME}"
# This script loops through the frameworks embedded in the application and
# removes unused architectures.
find "$APP_PATH" -name '*.framework' -type d | while read -r FRAMEWORK
do
FRAMEWORK_EXECUTABLE_NAME=$(defaults read "$FRAMEWORK/Info.plist" CFBundleExecutable)
FRAMEWORK_EXECUTABLE_PATH="$FRAMEWORK/$FRAMEWORK_EXECUTABLE_NAME"
echo "Executable is $FRAMEWORK_EXECUTABLE_PATH"
EXTRACTED_ARCHS=()

                        for ARCH in $ARCHS
                        do
                        echo "Extracting $ARCH from $FRAMEWORK_EXECUTABLE_NAME"
                        lipo -extract "$ARCH" "$FRAMEWORK_EXECUTABLE_PATH" -o
"$FRAMEWORK_EXECUTABLE_PATH-$ARCH"
                           EXTRACTED_ARCHS+=("$FRAMEWORK_EXECUTABLE_PATH-$ARCH")
                        done

                        echo "Merging extracted architectures: ${ARCHS}"
                        lipo -o "$FRAMEWORK_EXECUTABLE_PATH-merged" -create
"${EXTRACTED_ARCHS[@]}"
                        rm "${EXTRACTED_ARCHS[@]}"
                        echo "Replacing original executable with thinned version"
                        rm "$FRAMEWORK_EXECUTABLE_PATH"
                             mv "$FRAMEWORK_EXECUTABLE_PATH-merged"
"$FRAMEWORK_EXECUTABLE_PATH"
                        done
```



STEP 8 : BUILD & RUN YOUR PROJECT

iOS

Build and run your project. Et voilà!

You are now watching a VR cast of an Intel® Sports event in iOS

REQUIREMENTS FOR ANDROID

Android Studio updated version - press here to acquire

Minimum Intel® supported Android™ version - Android 4.4 Kit Kat (API Level 19) and above

Intel® **Immersive Experience** SDK framework kit archive

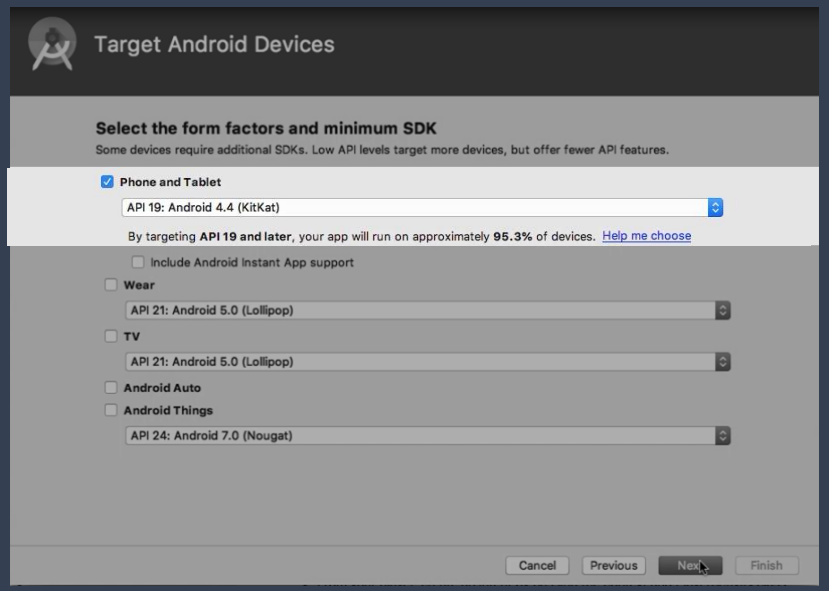
Overall procedure time: ~ 6-10 min

* For the full guidance read User Manual_Android.doc

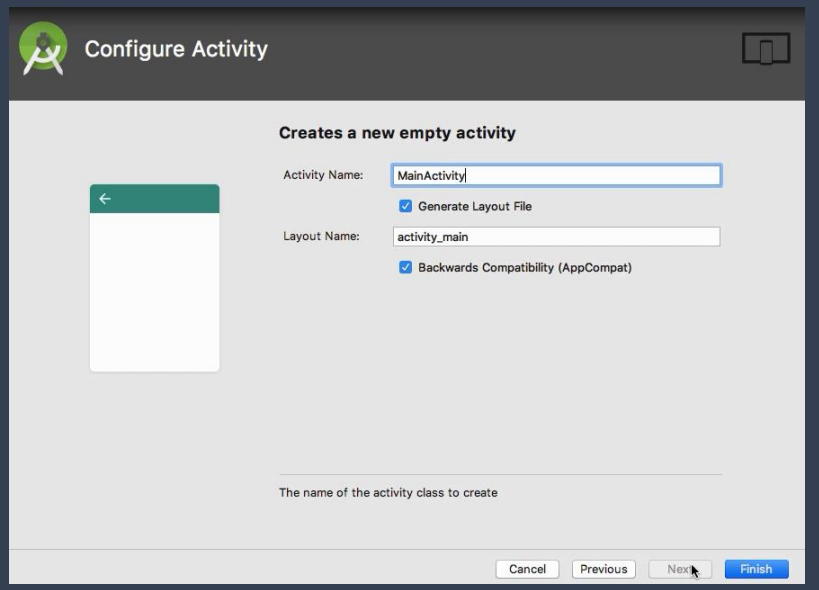
STEP 1 : SETUP PROJECT

Android

1. Create a new Android project
2. Select a Phone and Tablet on the Target Android Devices screen with the minimum SDK set to Android 4.4 Kit Kat



3. Add an Empty Activity and name it (say MainActivity)



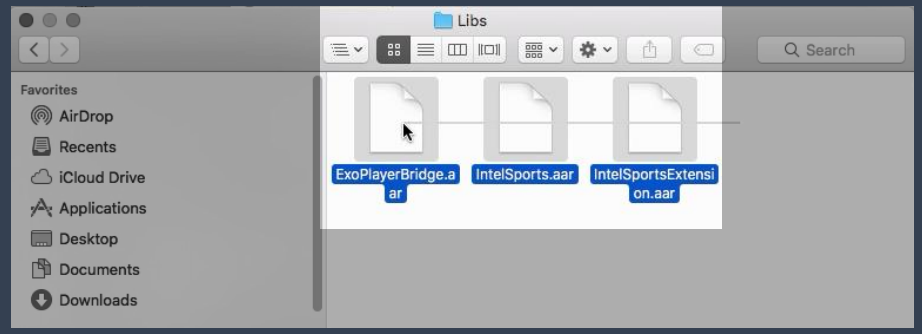
STEP 2 : ADD FRAMEWORK

Android

1 Browse and copy the .aar files:

ExoPlayerBridge.aar, IntelSports.aar and IntelSportsExtension.aar

from the provided Intel® Sports SDK kit > Libs folder

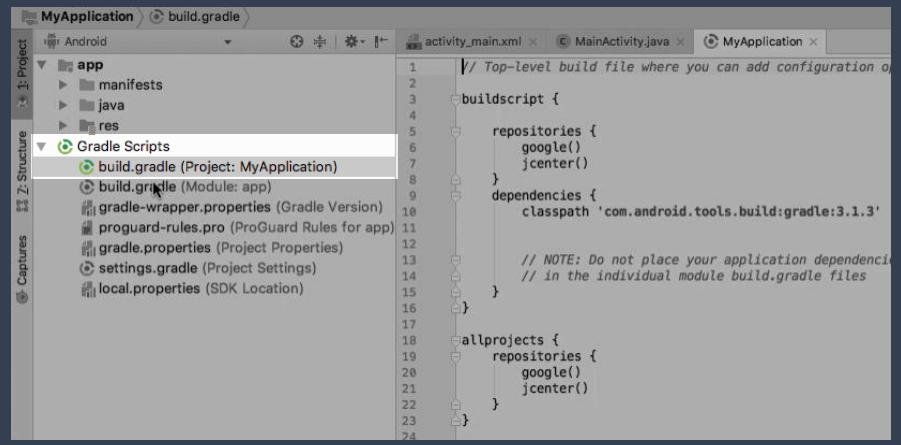


2 Paste the .aar files to the libs folder under your new app folder

Note:

Create a libs folder under your app folder, if not present

3 Open the root level build.gradle file

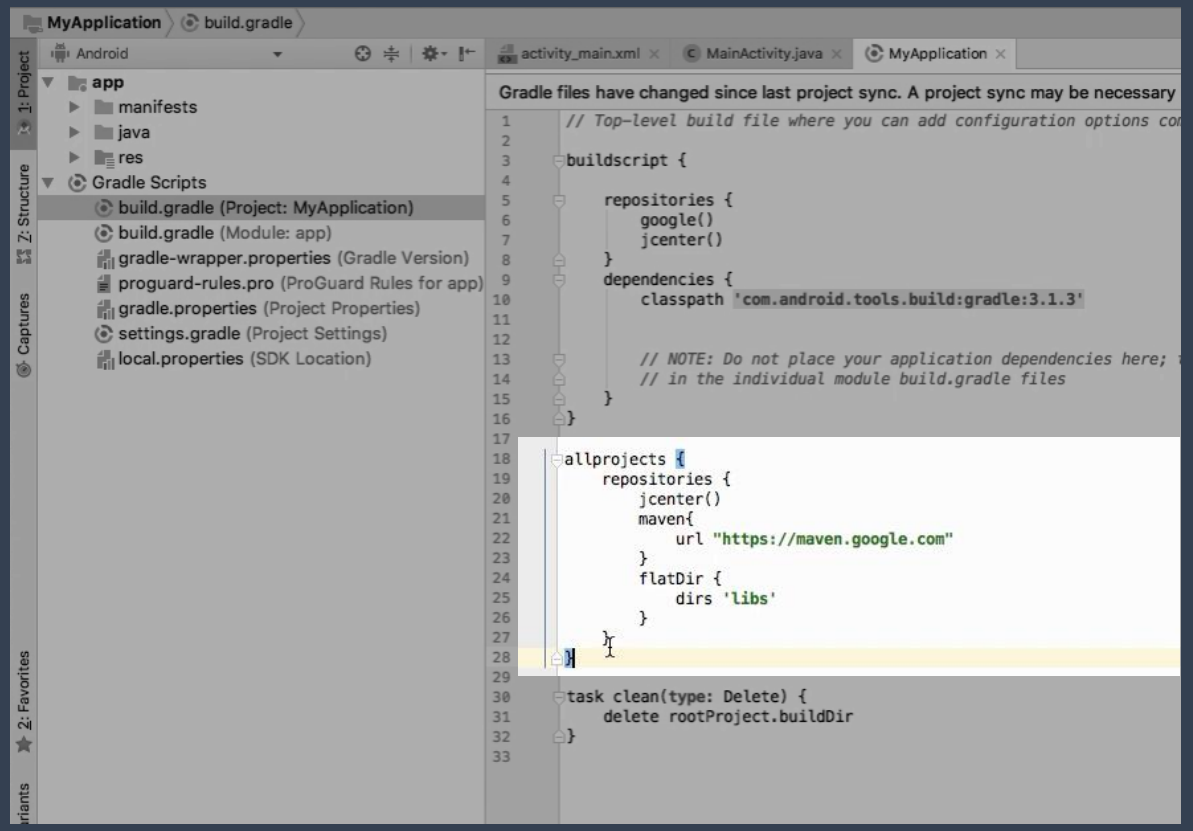


STEP 2 : ADD FRAMEWORK

Android

4 Copy the following 'allprojects' script and paste to the root level build.gradle (Project: MyApplication) file to replace the existing allprojects script

```
allprojects {
repositories {
 jcenter()
 maven{
  url "https://maven.google.com"
 }
 flatDir {
  dirs 'libs'
 }
}
}
```

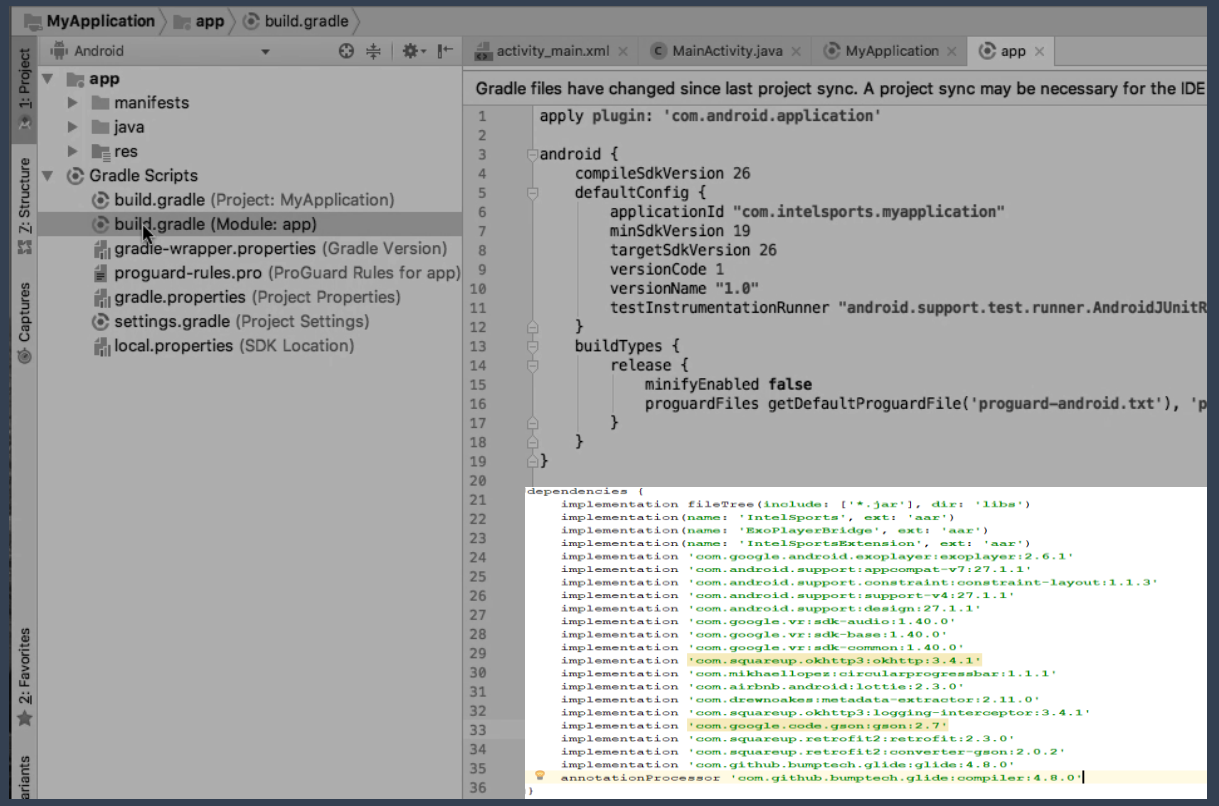


STEP 2 : ADD FRAMEWORK

Android

5 Copy the following 'dependencies' script and paste to the build.gradle (Module: app) file to replace the existing one

```
dependencies {
implementation fileTree(include: ['*.jar'], dir: 'libs')
   implementation(name: 'IntelSports', ext: 'aar')
   implementation(name: 'ExoPlayerBridge', ext: 'aar')
   implementation(name: 'IntelSportsExtension', ext: 'aar')
   implementation 'com.google.android.exoplayer:exoplayer:2.6.1'
   implementation 'com.android.support:appcompat-v7:27.1.1'
   implementation 'com.android.support.constraint:constraint-layout:1.1.3'
   implementation 'com.android.support:support-v4:27.1.1'
   implementation 'com.android.support:design:27.1.1'
   implementation 'com.google.vr:sdk-audio:1.40.0'
   implementation 'com.google.vr:sdk-base:1.40.0'
   implementation 'com.google.vr:sdk-common:1.40.0'
   implementation 'com.squareup.okhttp3:okhttp:3.4.1'
   implementation 'com.mikhaellopez:circularprogressbar:1.1.1'
   implementation 'com.airbnb.android:lottie:2.3.0'
   implementation 'com.drewnoakes:metadata-extractor:2.11.0'
   implementation 'com.squareup.okhttp3:logging-interceptor:3.4.1'
   implementation 'com.google.code.gson:gson:2.7'
   implementation 'com.squareup.retrofit2:retrofit:2.3.0'
   implementation 'com.squareup.retrofit2:converter-gson:2.0.2'
   implementation 'com.github.bumptech.glide:glide:4.8.0'
   annotationProcessor 'com.github.bumptech.glide:compiler:4.8.0'
}
```



Note:

Please note compileSdkVersion, targetSdkVersion and com.android.support need be same version.

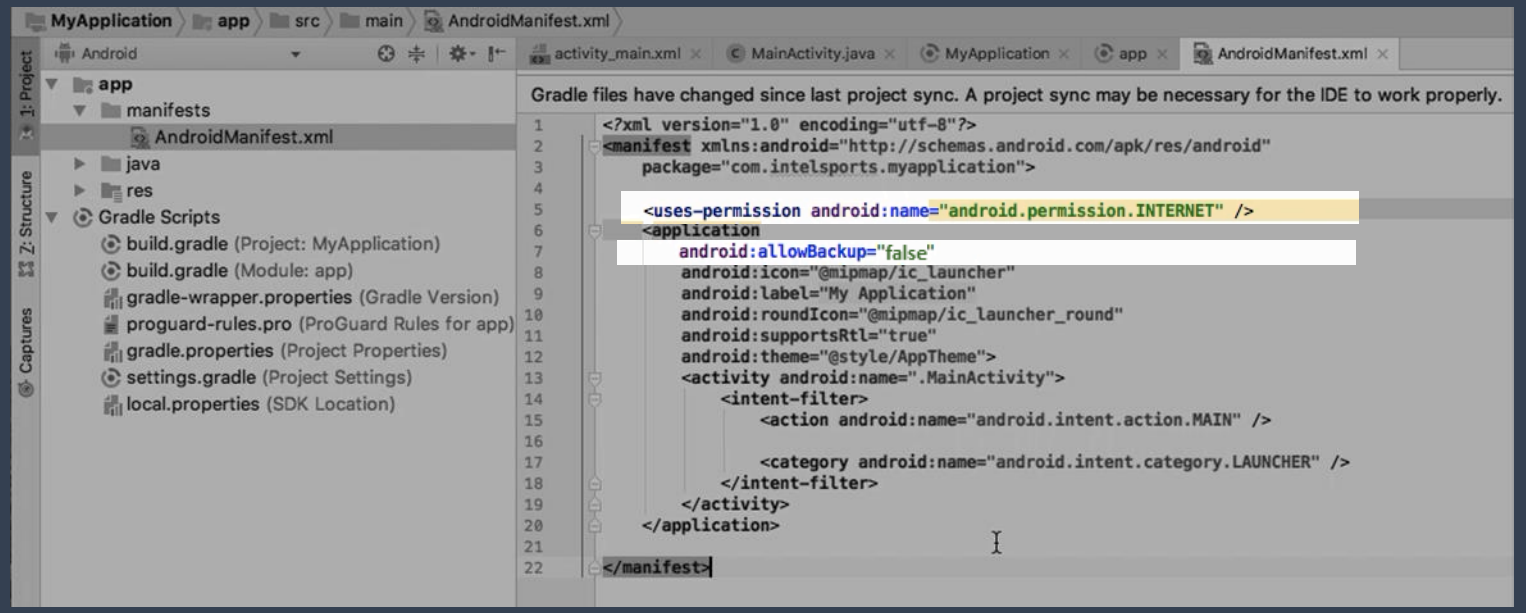
STEP 3 : UPDATE MANIFEST FILE

Android

1 Copy the following line and paste to the AndroidManifest.xml

```
<uses-permission android:name="android.permission.INTERNET" />
```

Note: Please set android:allowBackup in AndroidManifest.xml to "false"

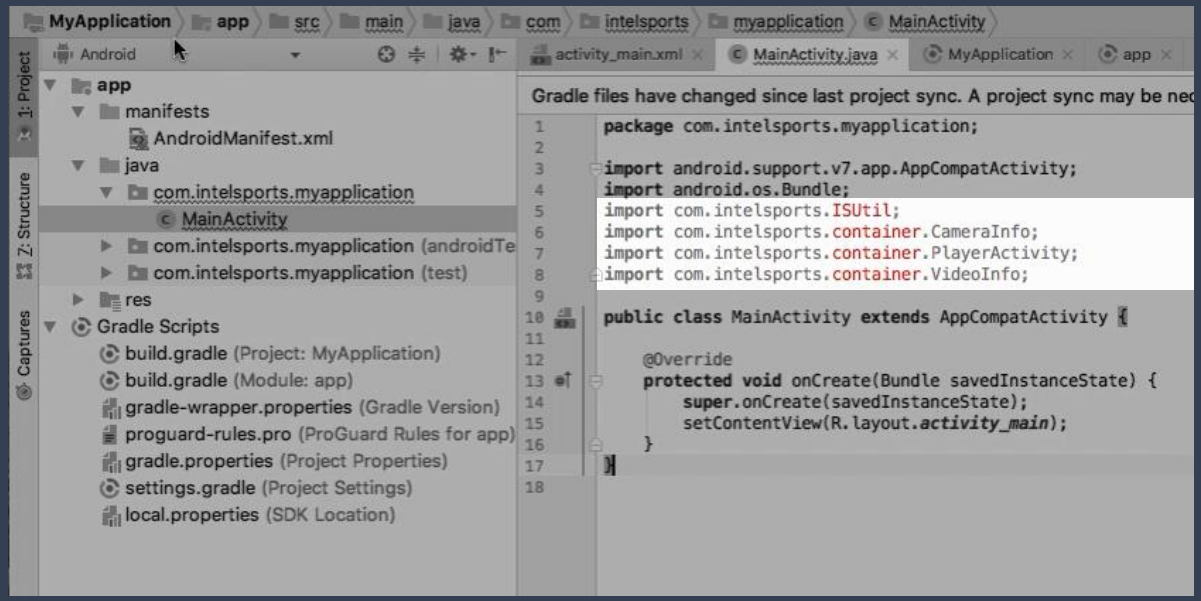


STEP 4 : CREATE THE VIDEO CONTROLLER

Android

1 Copy the following import packages lines and paste it to the MainActivity.java

```
import com.intelsports.ISUtil;
import com.intelsports.container.CameraInfo;
import com.intelsports.container.PlayerActivity;
import com.intelsports.container.VideoInfo;
import android.content.Intent;
```



STEP 4 : CREATE THE VIDEO CONTROLLER

Android

2 Copy the following play() function script including cameras URLs and paste to the MainActivity.java

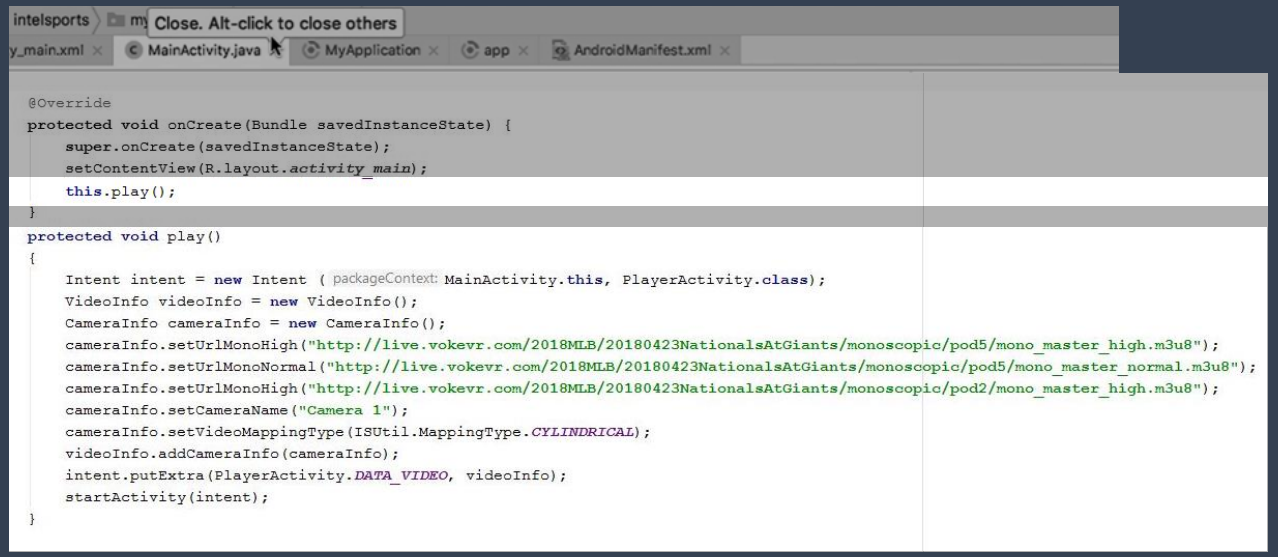
```
protected void play(){
  Intent intent = new Intent (MainActivity.this, PlayerActivity.class);

VideoInfo videoInfo = new VideoInfo();
CameraInfo cameraInfo = new CameraInfo();

cameraInfo.setUrlMonoHigh("http://live.vokevr.com/2018MLB/20180423NationalsAtGiants/monoscopic/pod5/mono_master_high.m3u8");
cameraInfo.setUrlMonoNormal("http://live.vokevr.com/2018MLB/20180423NationalsAtGiants/monoscopic/pod5/mono_master_normal.m3u8");
cameraInfo.setUrlMonoHigh("http://live.vokevr.com/2018MLB/20180423NationalsAtGiants/monoscopic/pod2/mono_master_high.m3u8");
cameraInfo.setCameraName("Camera 1");
cameraInfo.setVideoMappingType(ISUtil.MappingType.CYLINDRICAL);
videoInfo.addCameraInfo(cameraInfo);

intent.putExtra(PlayerActivity.DATA_VIDEO, videoInfo);
startActivity(intent);
}
```

3 Add `this.play();` to your onCreate(Bundle savedInstance) function



EXTRA : CAMERA SWITCHING

Android

* For the extra Camera Switching functionality, which enables end user to switch between the video streams, load up more CameraInfo URLs to the play() function

Note:

In panoramic viewing mode, the SDK will render the top image of a stereoscopic top/bottom video source, if a monoscopic URL is not passed to the CameraInfo class as a separate video source.

MonoHigh (Low Home camera):
http://live.vokevr.com/2018MLB/20180423NationalsAtGiants/monoscopic/pod2/mono_master_high.m3u8

MonoNormal (Low Home camera):
http://live.vokevr.com/2018MLB/20180423NationalsAtGiants/monoscopic/pod2/mono_master_normal.m3u8

MonoHigh (First Base camera):
http://live.vokevr.com/2018MLB/20180423NationalsAtGiants/monoscopic/pod1/mono_master_high.m3u8

MonoNormal (First Base camera):
http://live.vokevr.com/2018MLB/20180423NationalsAtGiants/monoscopic/pod1/mono_master_normal.m3u8

MonoHigh (Third Base camera):
http://live.vokevr.com/2018MLB/20180423NationalsAtGiants/monoscopic/pod3/mono_master_high.m3u8

MonoNormal (Third Base camera):
http://live.vokevr.com/2018MLB/20180423NationalsAtGiants/monoscopic/pod3/mono_master_normal.m3u8

MonoHigh (Dug Out camera):
http://live.vokevr.com/2018MLB/20180423NationalsAtGiants/monoscopic/pod4/mono_master_high.m3u8

MonoNormal (Dug Out camera):
http://live.vokevr.com/2018MLB/20180423NationalsAtGiants/monoscopic/pod4/mono_master_normal.m3u8

STEP 5: CREATE INTERFACE FOR THE MOMENTS CONTENT

Android

1 For enabling the volumetric clips and panoramic point of view content, which is based on the highlights moments , copy the following import packages lines to the MainActivity.java:

```
import com.intelsports.momentview.ISMomentActivity;
import com.intelsports.momentview.models.Moment;
import com.intelsports.momentview.models.Player;
import java.util.UUID;
```

2 From your player, copy the following lines to create an intent into the play() function:

```
Intent intentMoment = new Intent (MainActivity.this, ISMomentActivity.class);
```

3 Copy the following lines into the play() function to create a Moment object, and additional functions to set the content parameters:

```
final Moment momentClips = new Moment();
momentClips.setId(UUID.randomUUID());
momentClips.setContextClipUrl("https://s3-us-west-2.amazonaws.com/mobile-sdk-test-content/nfl/true_view/ISTV_WK5_IND_NE_17_CC_360_230318_360/AVC/mono_master_high.m3u8");
momentClips.setWauwClipUrl("https://s3-us-west-2.amazonaws.com/live-dev.content.sports.intel.com/nfl-mobile-test/360_WAUW_mono.mp4");
{
Player player1 = new Player("Brandon Graham", 55, UUID.randomUUID(), "Defensive End", "JAGUARS",
 "http://static.nfl.com/static/content/public/static/img/fantasy/transparent/200x200/GRA130064.png",
Player.BTP_TYPE.btp1FV,
 "https://s3-us-west-2.amazonaws.com/mobile-sdk-test-content/nfl/patriots.jpg",
 "https://s3-us-west-2.amazonaws.com/mobile-sdk-test-content/nfl/49ers.jpeg");

momentClips.addPlayer(player1);
}
{
Player player2 = new Player("Nate Solder", 55, UUID.randomUUID(), "Tight End","JAGUARS",
"http://static.nfl.com/static/content/public/static/img/fantasy/transparent/200x200/SOL076905.png",
Player.BTP_TYPE.btp1FV,
"https://s3-us-west-2.amazonaws.com/mobile-sdk-test-content/nfl/patriots.jpg",
 "https://s3-us-west-2.amazonaws.com/mobile-sdk-test-content/nfl/49ers.jpeg");
   momentClips.addPlayer(player2);
}
{
Player player3 = new Player("David Andrews", 87, UUID.randomUUID(), "Tight End","JAGUARS",
"http://static.nfl.com/static/content/public/static/img/fantasy/transparent/200x200/GRO135948.png",
Player.BTP_TYPE.btp1FV,
"https://s3-us-west-2.amazonaws.com/mobile-sdk-test-content/nfl/patriots.jpg",
"https://s3-us-west-2.amazonaws.com/mobile-sdk-test-content/nfl/49ers.jpeg");
   momentClips.addPlayer(player3);
}
```

STEP 5: CREATE INTERFACE FOR THE MOMENTS CONTENT

Android

4 For starting the moment activity, copy the following lines into the play() function:

```
intentMoment.putExtra(ISMomentActivity.DATA_MOMENT, momentClips);
startActivity(intentMoment);
```

STEP 6 : ADDING A NATIVE CONTAINER FOR LISTING INTERACTIVE REPLAY LIST

Android

For starting the native container activity

1 Copy the following lines into the file from which to launch the native container activity:

```
Import com.intelsports.momentview.container.MomentContainerActivity;
```

2 Set the CMS URL, CMS credential and CDN credential to the native container Activity intent, then start the activity, below are the sample codes:

```
Intent intent = new Intent(MainActivity.this, MomentContainerActivity.class);
        intent.putExtra(MomentContainerActivity.DATA_MOMENT_CONTAINER_CMS_URL, "<YOUR_CMS_URL>" );
        intent.putExtra(MomentContainerActivity.DATA_MOMENT_CONTAINER_CMS_USER, "<YOUR_USER_NAME>");
        intent.putExtra(MomentContainerActivity.DATA_MOMENT_CONTAINER_CMS_PASSWORD, "<YOUR_PASSWORD>");
        intent.putExtra(MomentContainerActivity.DATA_MOMENT_CONTAINER_CDN_CLIENTID, "<YOUR_CLIENTID>");
        intent.putExtra(MomentContainerActivity.DATA_MOMENT_CONTAINER_CDN_SECRET, "<YOUR_CDN_PASSWORD_with_base64_encode>");
        intent.putExtra(MomentContainerActivity.DATA_MOMENT_CONTAINER_CDN_DURATION, 300000L);

        startActivity(intent);
```

Note: The required parameters to support these implementations will be provided by Intel using key management access control methods. Please contact your Intel developer relations representative.

Note: The recommended access duration in the code snippet is set to 5 minutes (300 x 1000ms). This parameter should be set per your implementation requirements.

STEP 7 : ADD THE PROGUARD RULES INTO THE APP'S PROGUARD FILE

Android

```
-keep class com.google.vr.** { *; }
-keep class com.google.vr.sdk.base.** { *; }
-keep class com.google.vrtoolkit.cardboard.** { *; }
-keep class com.google.protobuf.nano.** { *; }
-keep class com.google.common.logging.nano.** { *; }
-dontwarn org.w3c.dom.bootstrap.DOMImplementationRegistry
#if Compile Android API <= 27
-dontwarn com.bumptech.glide.load.resource.bitmap.VideoDecoder

-keep class com.intelsports.ISVideoView {
        void OnEnteringToScene(java.lang.String);
         void OnExitFromScene(java.lang.String);
 }
-keepclasseswithmembernames,includedescriptorclasses class * {
     native <methods>;
 }
-keepclassmembers class * implements android.os.Parcelable {
    static ** CREATOR;
}
-keep class com.google.android.exoplayer.** {*;}

# Retrofit ----------------------------------------------START
# Platform calls Class.forName on types which do not exist on Android to determine platform.
-dontnote retrofit2.Platform
# Platform used when running on Java 8 VMs. Will not be used at runtime.
-dontwarn retrofit2.Platform$Java8
# Retain generic type information for use by reflection by converters and adapters.
-keepattributes Signature
# Retain declared checked exceptions for use by a Proxy instance.
-keepattributes Exceptions
# For Okie
-dontwarn okio.**
# For OkHttp
-dontwarn okhttp3.**
-dontwarn okio.**
-dontwarn javax.annotation.**
# Retrofit ----------------------------------------------END
```

STEP 8: BUILD & RUN YOUR PROJECT

Android

Et voilà!

You are now watching a VR cast of an Intel® Sports event in Android

COPYRIGHT 2017-2019 INTEL® SPORTS

TRADEMARK NOTICES

Android™ is a trademark of Google, Inc.
iOS™ is a trademark of Apple, Inc.
All other trademarks are the property of their respective owners.