

Android & iOS Native Screen Recorder

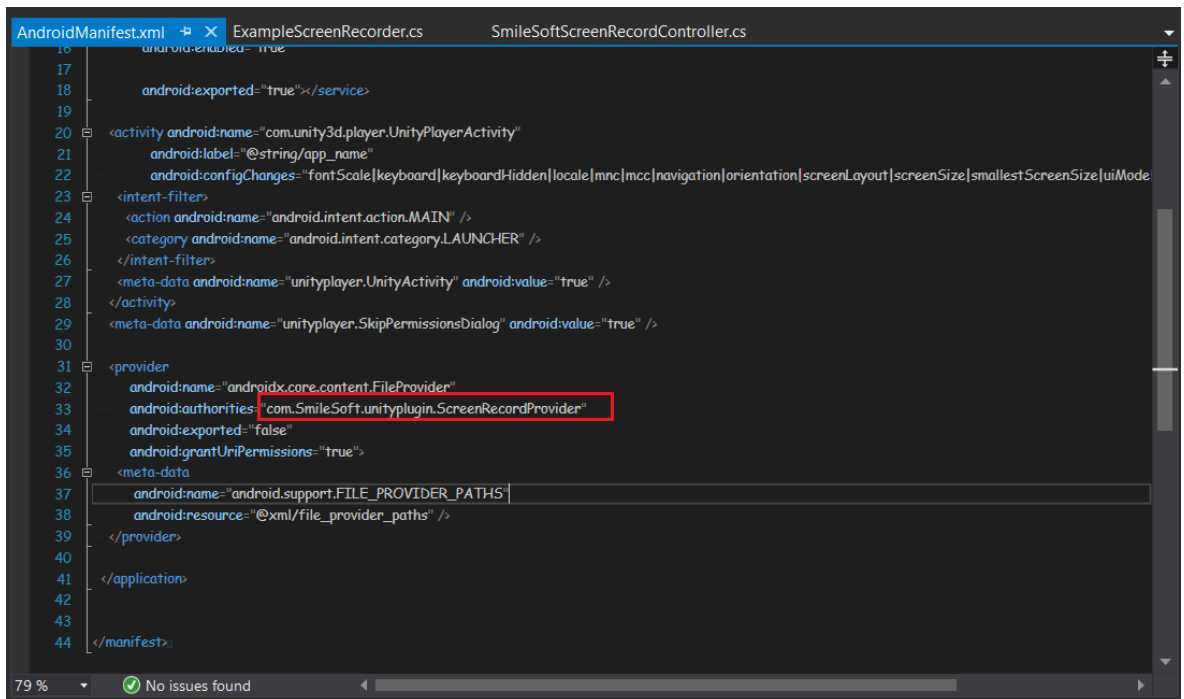
This plugin can record your gameplay without any issues. It uses Android native **MediaRecorder** API and **Android Foreground Service** for the record screens. So it will support from **Lollipop (5.1)** to newer versions even **Android 13(The latest One)** and above.

In iOS, it uses the iOS replayKit Library. It will support all iPhone, iPad, and iPod Touch devices with iOS 9.0 or above.

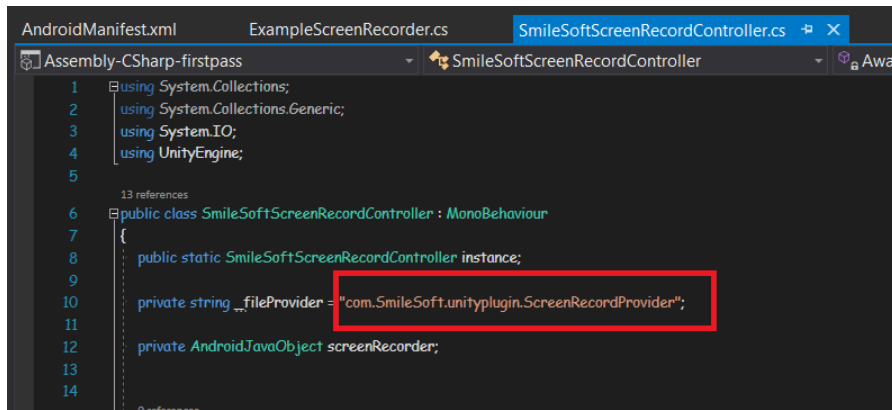
Instruction: After importing the project please drag **Screen Recorder** prefab in Scene Hierarchy from **Assets > SunShine Android Native Screen Recorder > Prefab > Screen Recorder**.

If you want to share the video using the native share dialog then please do the following settings. **(Only Android)**

1. Open the Android manifest file from **Assets > Plugins > Android > AndroidManifest.xml**. Now change the authority name with a unique one. We highly recommend you use the package name.



2. Again Open **SmileSoftScreenRecordController.cs** from **Assets > SunShine Android Native Screen Recorder > Scripts > SmileSoftScreenRecordController.cs**. Now change the `_fileProvider` value which you set on the manifest file before.



```
1 using System.Collections;
2 using System.Collections.Generic;
3 using System.IO;
4 using UnityEngine;
5
6 public class SmileSoftScreenRecordController : MonoBehaviour
7 {
8     public static SmileSoftScreenRecordController instance;
9     private string _fileProvider = "com.SmileSoft.unityplugin.ScreenRecordProvider";
10    private AndroidJavaObject screenRecorder;
11
12
13
14
```

You can also see this [video](#) For setting the file provider path value.

Now you are ready for calling the plugin API. The following functions are available in this plugin.

Functions:

1. **Start Record:** `SmileSoftScreenRecordController.instance.StartRecording()`
2. **Stop Record:** `SmileSoftScreenRecordController.instance.StopRecording()`. In Android, It returns the recorded file path. in iOS devices, it will open the native preview window. From there you can edit, discard or save the recorded video in the camera roll even you can share it instantly with your sharing apps.
3. **Record Audio:** `SmileSoftScreenRecordController.instance.SetAudioCapabilities(isAudioRecording)`. Where **isAudioRecording** is a **Boolean** type parameter. This will record audio from the device mic.
4. **Set File Destination:** `SmileSoftScreenRecordController.instance.SetVideoDestination(destination)`. Where the **destination** is a **String** type value. By default, the destination is the **External Storage** directory. **(Only Android)**
5. **Set File Name:** `SmileSoftScreenRecordController.instance.SetVideoName(fileName)`. Where **fileName** is a **String** type value. **(Only Android)**
6. **Folder Name:** You can create a new folder and save your video there. Just call `SmileSoftScreenRecordController.instance.SetStoredFolderName(folderName)` function. Here **folderName** is a **String** type value. **(Only Android)**
7. **Set Gallery Add Capabilities:** `SmileSoftScreenRecordController.instance.SetGalleryAddingCapabilities(canAddIntoGallery)`. Where **canAddintoGallery** is a **boolean** type value. If it is true then the video file will be added to the device gallery. By default this value is **true**. **(Only Android)**
8. **Bit Rate:** `SmileSoftScreenRecordController.instance.SetBitRate(bitrate)`. **Bitrate** is an **integer-type** parameter. A higher bitrate means high-quality video. Please try to set this value like - a * 512 * 512. change the value of a. **(Only Android)**
9. **Preview:** If you want to see the preview in the native video view then just call this function `SmileSoftScreenRecordController.instance.PreviewVideo(_recordedFilePath)`. Here **recordedFilePath** is a **string-type** parameter. For iOS, you do not have to pass the file path.

10. **Share Video:** You can also Share the video using the native share dialog. Just call `SmileSoftScreenRecordController.instance.ShareVideo(_recordedFilePath, message, share title)`. It takes three parameters and all of them are **strings**. The first one is the **filePath**, the second one is the **share message** and the third one is the **share title**. **(Only Android)**
11. **Video Size:** `SmileSoftScreenRecordController.instance.SetVideoSize(width, height)`. Both **width** and **height** are integer-type parameters. **(Only Android)**
12. **Video Encoder:** `SmileSoftScreenRecordController.instance.SetVideoEncoder(videoEncoder)`. Here **video encoder** is an integer-type parameter. Before setting this please look at the Android [Developer official](#) site for supporting encoders in different Android API versions. **(Only Android)**
13. **Video Rotation:** `SmileSoftScreenRecordController.instance.SetVideoRotation(Videorotation)`. Here **video rotation** is an integer-type parameter. Values should be 0, 90, 180, or 270. Before setting this please look at the Android [Developer official](#) site for supporting rotations in different encoders and different Android API versions. **(Only Android)**

To get a clear idea please see the example scene from **Assets > SunShine Android Native Screen Recorder > Example > Example Scene**. As

**** Here 3-9 number functions are optional. You can use all of them or none. But please be sure that these (3-9) functions are called before calling the Start Record function. ****

Android Callback Functions: You get the following callbacks in `SmileSoftScreenRecordController.cs` script.

OnRecordPermissionGranted - It will call when the Record permission popup opens and users accept/reject the permission.

Parameter: Status (string)

True - Accept permission.

False - Declined Permission

OnRecordStartStatus - It will call when a user tries to record.

Parameter: Status (string)

True - Record Start Successfully

False - Record Failed

iOS Callback Functions: You get the following callbacks in `SmileSoftScreenRecordController.cs` script.

OnIosRecordStatus - It will call when the iOS Record will start or fail.

Parameter: Status (Boolean)

True - Successfully start recording

False - Failed to start recording

Troubleshoot -

Android - If you show the duplicate library error during build then please download the External [Dependency Manager](#) plugin.

DO the manual resolution using the following menu options:

- Assets > External Dependency Manager > Android Resolver > Resolve
- Assets > External Dependency Manager > Android Resolver > Force Resolve

IOS - In iOS only **file path** and **message** parameters are supported. Others are set automatically. So you do not have to worry about other parameters.

In iOS you might see a (**dyld: Library not loaded: @rpath/libswiftCore.dylib**) error when you try to build it in an iOS device. For resolving this from Xcode set **Always Embed Swift Standard Libraries** value to **true**. You found this in the **Build Setting** tab.

