## **Native Share For Android And iOS**

With a single line of code, you can share any single or multiple files from the Unity application. Setup is very simple and easy.

Project Setup: Just Drag and drop the **Native Share** prefab into the game hierarchy. You will find this prefab in **Assets/Plugins/SunShine Native Share/Prefab/Native Share**. Now you are ready to call the plugin API.

For iOS it is all but for Android, you need to do one thing more. Please do the following steps for the Android platform only.

Open androidManifest.xml file from Plugins / Android / androidManifest.xml. Replace the android:authorities name from the provider block with some unique name. This will be your file provider path.

```
× 7 Androidhlanifectumi
      * SunihineHallvelhars.co × Testifians.co
Assembly-CSharp > manifest > application > provider > @android.eathorities.
       1 chml version="1.8" encoding="utf-8"3>
3 cmanifest umlns:android="http://schemps.android.com/apk/res/android"
                     mains:teels="http://schemas.android.com/toels"
package="com.SedieSoft.undtyplugin">
             capplication android:label="gstring/app_name" android:icon="gdrawable/app_icon" >
            cactivity and reldinance" con. unity10.player.UnityFlayerActivity" and reldilabel="forting/app.ness" and reldispressful Sensor" and reldilabels.
                  cartion android-manes android-intent-action-NATA />
              icategory android:mase" android.intent.category.LAMONES" />
category android:mase" android.intent.category.LAMONES" />

//intent-filter>
     11
     13
14
15
16
17
18
19
                 creta-data android:name-"unityplayer.unityActivity" android:value-"true" />
                  android:mame="android support of present FileDo
android:authorities="com.SelleSoft.anityplugin.
                                                                                ShareFrovider_test* replace this with Some unique name
     21
22
23
24
25
26
27
28
                    android:exported-"!
                      android:grantUriFernissions="true">
                    android:name="undroid.support.FILE_MOVIES_NAME"
android:nesource="post/file_provider_paths" />
               ouses-sdk android:min6dWersion*16" android:target5dWersion*27" />
      50 c/martifestis
```

Then Again open SunShineNativeShare.cs script from Assets/Plugins / SunShine Native Share / Scripts/ SunShineNativeShare.cs. Copy the provider path from androidmanifest file and paste it into FileProviderName variable.

Video tutorial for setup is here <a href="https://youtu.be/GuCw5plwxtl">https://youtu.be/GuCw5plwxtl</a>

```
    O ShareTest(string message, string shareDialogTitle)

        using UnityEngine;
      Epublic class SunShineMativeShare : MonoSehavdour
             public static string TVPE_IPAGE = "image/";
9
18
11
12
            private const string SMARE_PACKAGE_NAME = "com.SmileSeft.unityplugin";
private const string SMARE_CLASS_NAME = ".ShareFragment";
13
14
             primate const string TEXT_SAMME_METHED = "Sharefest";
15
16
17
            private const string SINGLE FILE SHAME METHOD - "ShareSingleFile";
18
19
28
21
            private const string fileProvidertame - "com.SmileSeft.unityplugin.ShareProvider_test";
                                                                                                                     Replace the provider path from manifest File
22
23
24
     public static void ShareText(string message, string shareDialogTitle)
25
26
27
                 using (Androidiavablect share android obj = new Androidiavablect(SHARE_PACKAGE_NAME + SHARE_CLASS_NAME))
                     share_android_obj.Call(TEXT_SHAME_METHOD, message, shareDialogTitle);
29
38
31
                 Debug.Log("Native Share just work in android Flatform");
32
33
34
35
36
             public static wold ShareSingleFile(string path, string fileType, string message, string shareBialogTitle)
39
48
41
                 using (AndroidlavaObject share_android_obj = new AndroidlavaObject(SHARE_PACKASE_NAME + SHARE_CLASS_NAME))
42
43
44
                      share_android_obj.Call(SINGLE_FILE_SHARE_METHOD, FileProviderName, path, fileType, message, shareDialogTitle);
45
                 Debug.Log("Native Share just work in android Flatform");
46
47
48
49
70
```

\*\* Strongly recommended that you should use your package name as your provider name \*\*

- **1. Share Text:** SunShineNativeShare.instance.ShareText (**string** message, **string** shareDialog)
- **2. Share Single File**: SunShineNativeShare. instance.ShareSingleFile( **string** path, **string** fileType, **string** message, **string** shareDialogTitle )
- 3. Share Multiple File of Same Type:

SunShineNativeShare.instance.ShareMultipleFileOfSameType ( string[] path , string fileType, string message, string shareDialogTitle )

4. Share Multiple File of Multiple Type: SunShineNativeShare. Instance.

ShareMultipleFileOfMultipleType ( **string**[] path , **string** message, **string** shareDialogTitle )

## 5. Share Callbacks (Android Only):

Check the ShareCallback function from SunShineNativeShare.cs script from Assets > Plugins > SunShine Native Share > scripts > SunShineNativeShare.cs

This function provides a string variable named **isSuccess**.

\*This callback behaves differently for different apps. So it is not guaranteed the success or failure status accurately. But this callback will call whenever the share dialog disappears.\*

## **Troubleshoot -**

**Android** - If you show the duplicate library error during build then please download the External <u>Dependency Manager</u> 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.

