

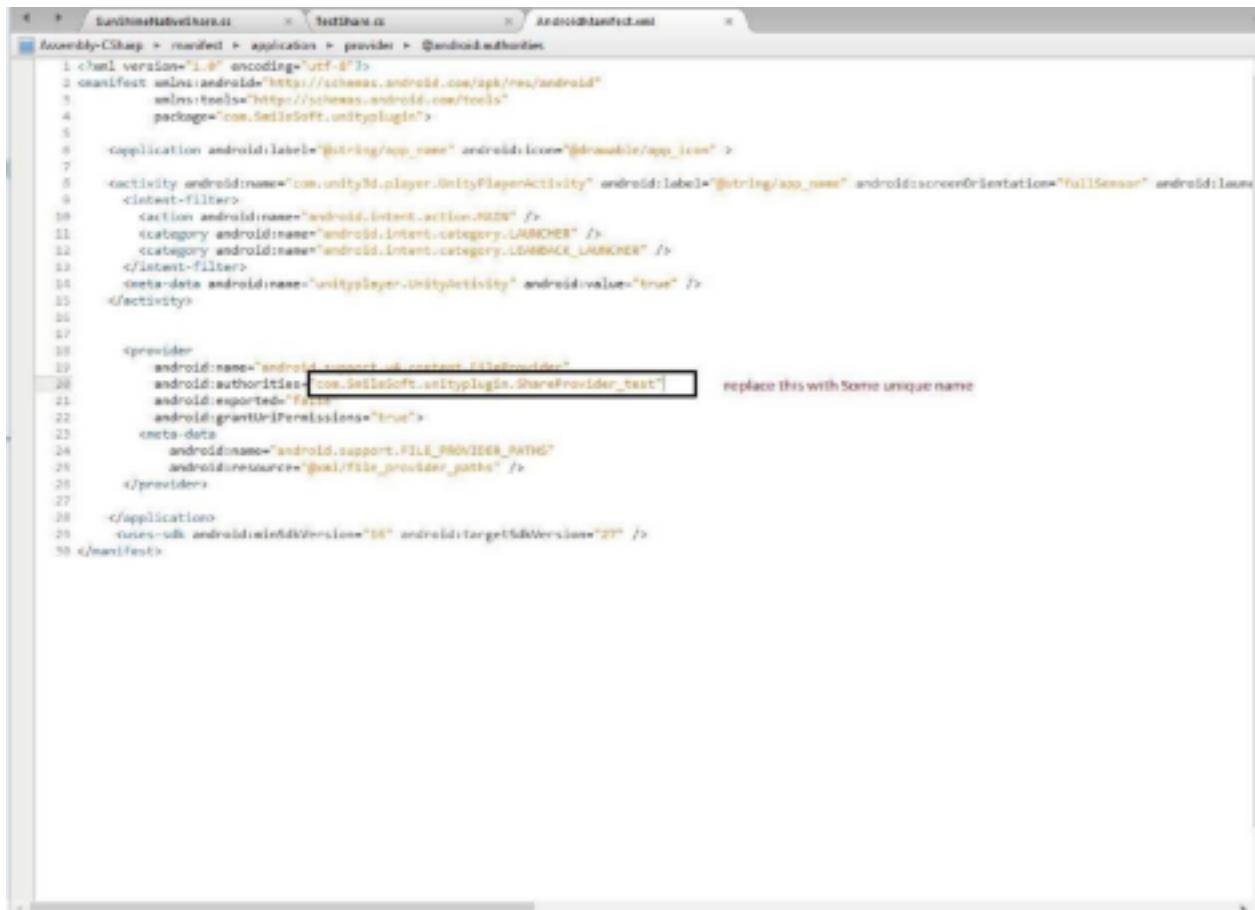
Native Share For Android And iOS

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.



```
1 <?xml version="1.0" encoding="utf-8"?>
2 <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3   xmlns:tools="http://schemas.android.com/tools"
4     package="com.Selladsoft.unityplugin">
5
6   <application android:label="@string/app_name" android:icon="@drawable/app_icon" >
7
8     <activity android:name=".UnityPlayerActivity" android:label="@string/app_name" android:screenOrientation="fullSensor" android:laun-
9       chMode="filter">
10       <action android:name="android.intent.action.MAIN" />
11       <category android:name="android.intent.category.LAUNCHER" />
12       <category android:name="android.intent.category.LEANBACK_LAUNCHER" />
13     </activity>
14
15
16     <provider
17       android:name="android.support.v4.content.FileProvider"
18       android:authorities="com.Selladsoft.unityplugin.ShareProvider_test" replace this with Some unique name
19       android:exported="false"
20       android:grantUriPermissions="true">
21       <meta-data
22         android:name="android.support.FILE_PROVIDER_PATHS"
23         android:resource="@xml/file_provider_paths" />
24     </provider>
25
26   </application>
27   <uses-sdk android:minSdkVersion="16" android:targetSdkVersion="21" />
28 </manifest>
```

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 <https://youtu.be/GuCw5plwxI>

```
SunshineNativeShares 4 X | TestShares
+ Unity Native Share Free Plugins
  + SunShareNativeShare
    + ShareTest(string message, string shareDialogTitle)
  + public class SunShareNativeShare : MonoBehaviour
  + {
  +     public static string TYPE_IMAGE = "image/*";
  + 
  +     private const string SHARE_PACKAGE_NAME = "com.SelleSoft.unityplugin";
  +     private const string SHARE_CLASS_NAME = ".ShareFragment";
  +     private const string TEXT_SHARE_METHOD = "ShareText";
  +     private const string SINGLE_FILE_SHARE_METHOD = "ShareSingleFile";
  + 
  +     private const string fileProviderName = "com.SelleSoft.unityplugin.ShareProvider_1015"; Replace the provider path from manifest file
  + 
  +     public static void ShareText(string message, string shareDialogTitle)
  +     {
  +         #if UNITY_ANDROID
  +             using (AndroidJavaObject share_android_obj = new AndroidJavaObject(SHARE_PACKAGE_NAME + SHARE_CLASS_NAME))
  +             {
  +                 share_android_obj.Call(TEXT_SHARE_METHOD, message, shareDialogTitle);
  +             }
  +         #endif
  +         Debug.Log("Native Share just work in android Platform");
  +     }
  + 
  +     public static void ShareSingleFile(string path, string fileType, string message, string shareDialogTitle)
  +     {
  +         #if UNITY_ANDROID
  +             using (AndroidJavaObject share_android_obj = new AndroidJavaObject(SHARE_PACKAGE_NAME + SHARE_CLASS_NAME))
  +             {
  +                 share_android_obj.Call(SINGLE_FILE_SHARE_METHOD, fileProviderName, path, fileType, message, shareDialogTitle);
  +             }
  +         #endif
  +         Debug.Log("Native Share just work in android Platform");
  +     }
  + }
```

**** 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**.

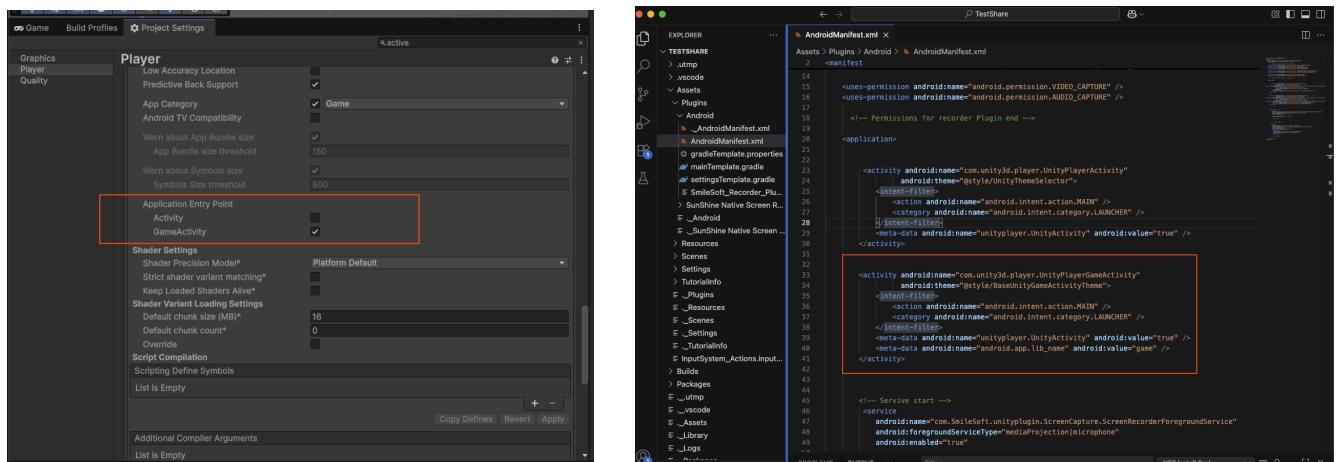
This callback behaves differently for different apps. So it is not guaranteed the success or failure status accurately. But this callback will call whenever the share dialog disappears.

Troubleshoot -

Android -

1. Check the Application Entry point from **Project Settings -> Select Android -> Other Settings -> Application Entry Point**. Make sure **Game Activity** is selected.

If you don't want to keep the **Game Activity** as the entry point, then remove the UnityPlayerGameActivity section from the activity part.



2. 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.

