

Sunshine Android and iOS Native Share:

With a single line of code you can share any single or multiple file from unity application. Setup is very simple and easy.

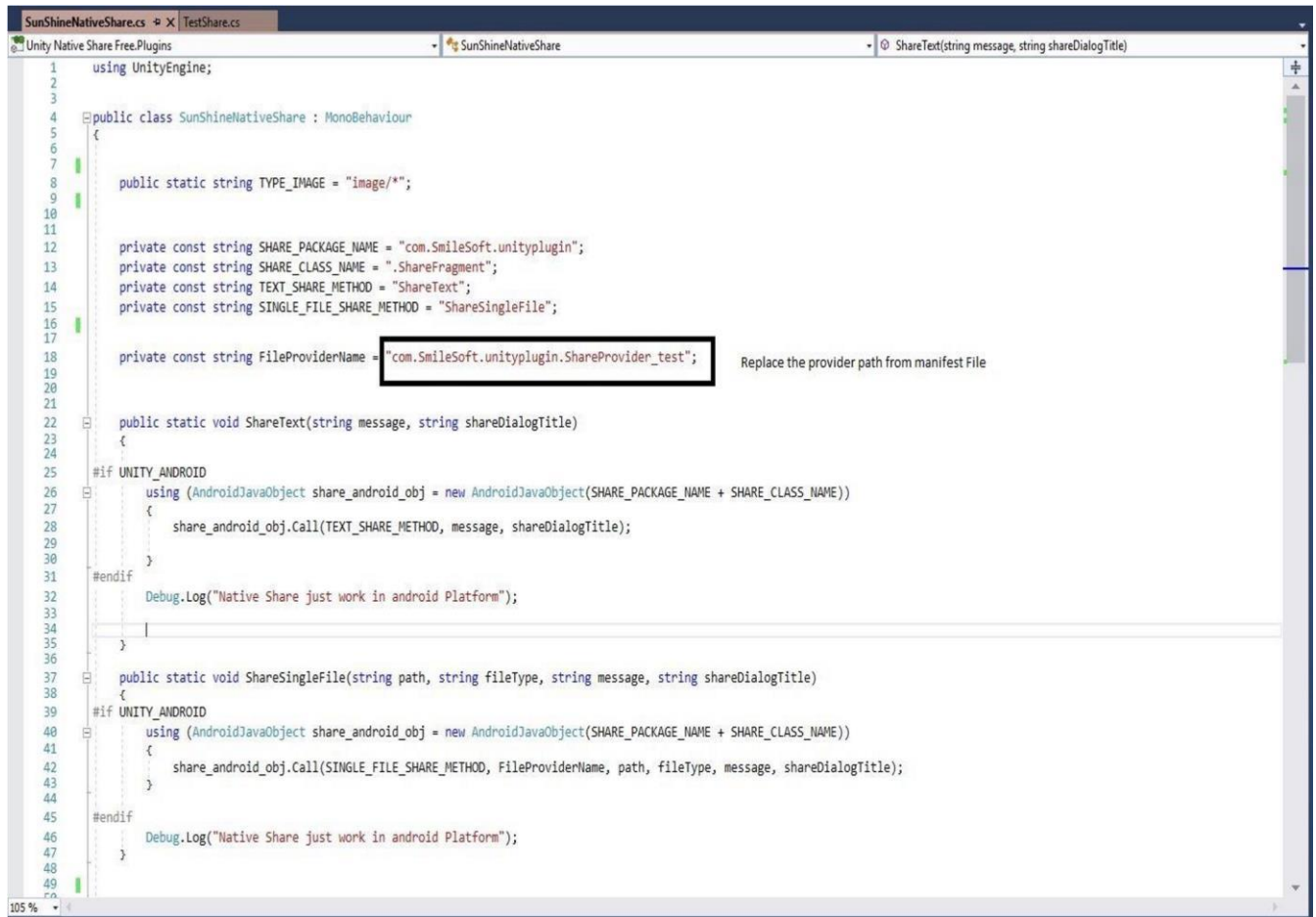
Project Setup: Just Drag and drop the **Native Share** prefab into game hierarchy. You will find this prefab in **Assets/Plugins/SunShine Native Share/Prefab/Native Share**. Now you are ready to call the plugin api.

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

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

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 in **FileProviderName** variable.

Video tutorial for setup is here <https://youtu.be/GuCw5plwxtI>



```
1 using UnityEngine;
2
3
4 public class SunShineNativeShare : MonoBehaviour
5 {
6
7     public static string TYPE_IMAGE = "image/*";
8
9
10
11     private const string SHARE_PACKAGE_NAME = "com.SmileSoft.unityplugin";
12     private const string SHARE_CLASS_NAME = ".ShareFragment";
13     private const string TEXT_SHARE_METHOD = "ShareText";
14     private const string SINGLE_FILE_SHARE_METHOD = "ShareSingleFile";
15
16
17     private const string FileProviderName = "com.SmileSoft.unityplugin.ShareProvider_test"; // Replace the provider path from manifest File
18
19
20
21
22     public static void ShareText(string message, string shareDialogTitle)
23     {
24
25         #if UNITY_ANDROID
26         using (AndroidJavaObject share_android_obj = new AndroidJavaObject(SHARE_PACKAGE_NAME + SHARE_CLASS_NAME))
27         {
28             share_android_obj.Call(TEXT_SHARE_METHOD, message, shareDialogTitle);
29         }
30         #endif
31         Debug.Log("Native Share just work in android Platform");
32     }
33
34
35
36
37     public static void ShareSingleFile(string path, string fileType, string message, string shareDialogTitle)
38     {
39         #if UNITY_ANDROID
40         using (AndroidJavaObject share_android_obj = new AndroidJavaObject(SHARE_PACKAGE_NAME + SHARE_CLASS_NAME))
41         {
42             share_android_obj.Call(SINGLE_FILE_SHARE_METHOD, FileProviderName, path, fileType, message, shareDialogTitle);
43         }
44         #endif
45         Debug.Log("Native Share just work in android Platform");
46     }
47
48
49
50 }
```

**** 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)`

In iOS only **file path** and **message** parameter 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 **true**. You found this in **Build Setting** tab.

