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.

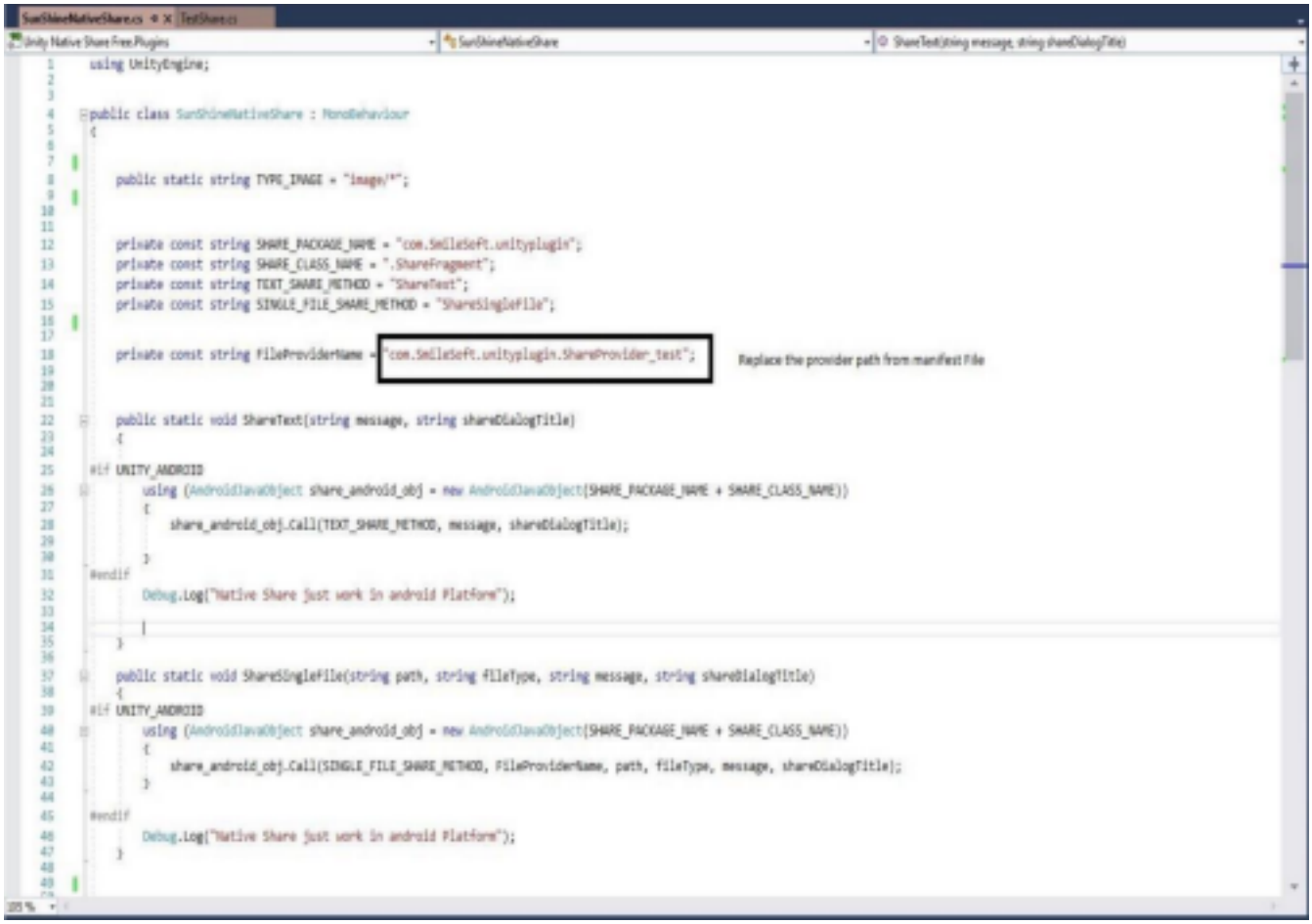
Project Setup: Just Drag and drop the **Native Share** prefab into the game hierarchy. You will find this prefab in **Assets/Plugins/SunShine Native Share/Prefab/Native Share**. Now you are ready to call the plugin API.

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.

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/GuCw5plwxtI



** 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.

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.