|  |  |
| --- | --- |
| 翻译前字数 | 669 |
| 目录 | Unity Manual/Platform-specific/Windows/Windows Store Apps/Scripting Backends/Windows Store: .Net Scripting Backend/Windows Store Apps: Plugins on .Net Scripting Backend |
| 链接 | https://docs.unity3d.com/540/Documentation/Manual/windowsstore-plugins.html |

**Windows Store Apps: Plugins on .NET Scripting Backend**

**Windows 商店应用: .Net后端脚本的插件**

**Managed Plugins**

**受托管的插件**

You cannot use Windows Store Apps specific plugins in Unity Editor if you use Windows Runtime APIs (<http://msdn.microsoft.com/en-us/library/windows/apps/br211377.aspx>), so we changed a bit how Unity Editor handles them. If you intend to use the plugin only for Windows Store Apps and not in Unity editor, you can skip making a placeholder. If you do this, you need to wrap the code which uses the plugin API with the following:

在UnityEditor中，您不可以使用Windows Store App专有插件的Windows Runtime API(<http://msdn.microsoft.com/en-us/library/windows/apps/br211377.aspx>)，稍作改动来让我们在Unity编辑器中使用他们。如果您仅为Windows 商店应用而不是在Unity编辑器中使用他们的话，您可以跳过制作一个placeholder。如果您制作placeholder，您需要将使用插件API的代码包装如下：

#if !UNITY\_EDITOR

// Plugin code

#endif

If you need a placeholder, you make two versions of plugins:

如果您需要一个placeholder，您需要制作两个版本的插件：

* One for Unity Editor
* 一个Unity编辑器的版本
* The other one for Windows Store (**Note:** We do not support Windows Phone 8.1 silverlight assemblies as plugins for Windows Phone 8.1)

**Important** The placeholder plugin for Unity Editor cannot reference UnityEditor.dll, otherwise you’ll get the following error:

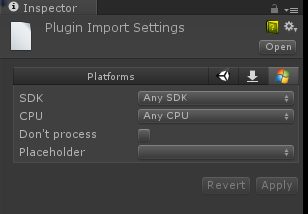
* 另一个是Windows商店的版本（注：我们不会为Windows Phone 8.1支持Windows Phone 8.1 silverlight的程序集）

**重要** 为Unity编辑器的placeholder插件不可以引用UnityEditor.dll，否则您将得到如下错误：

The Assembly UnityEditor is referenced by Plugin ('Assets/Plugins/Plugin.dll'). But the dll is not allowed to be included or could not be found.

Both of them must share the same name and have the same assembly version. For example, you should place Editor compatible plugin in Assets\Plugins\MyPlugin.dll, and Windows Store Apps specific plugin in Assets\Plugins\WSA\MyPlugin.dll.

他们两者都必须使用相同的名字并且程序集的版本相同。举个例子，您应该将编辑器兼容的版本放在Assets\Plugins\MyPlugin.dll，并且将Windows Store Apps专有的插件放在Assets\Plugins\WSA\MyPlugin.dll。

* Go to Assets\Plugins\MyPlugin.dll, select Editor as the only compatible platform
* 找到Assets\Plugins\MyPlugin.dll，选择编辑器作为唯一兼容的平台
* Go to Assets\Plugins\WSA\MyPlugin.dll, select Windows Store Apps as the only compatible platform, now go to Windows Store Apps plugin settings
* 找到Assets\Plugins\WSA\MyPlugin.dll，选择Windows Store Apps为唯一兼容的平台
* Pick ‘Assets\Plugins\MyPlugin.dll’ in the placeholder field, this means that when building to Windows Store apps ‘Assets\Plugins\MyPlugin.dll’ will be used when compiling your scripts, but ‘Assets\Plugins\WSA\MyPlugin.dll’ will be copied to final folder instead of ‘Assets\Plugins\MyPlugin.dll’. This achieves two things - Unity Editor will successfully compile your scripts, but during the game you’ll be using API from Windows Store Apps specific plugin.
* 选择‘Assets\Plugins\MyPlugin.dll’的placeholder字段，意思是当构建到Windows商店app时，‘Assets\Plugins\MyPlugin.dll’将编译您的脚本，但是‘Assets\Plugins\WSA\MyPlugin.dll’ 将被拷贝到最终目录替换‘Assets\Plugins\MyPlugin.dll’。实现了两件事 - Unity编辑器将成功编译您的脚本，但在游戏中您将使用从Windows商店app专有插件里的API。  
  

**Don’t process** option

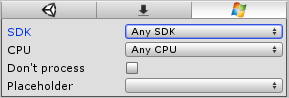
不处理选项

This option is used when you don’t want for Unity to patch your assemblies, usually this option should be applied for plugins which contain heavy Windows Runtime API, and Unity fails to patch them.

当您不想用Unity修补程序集时使用此选项，通常使用了大量Windows运行时的API时应用此选项并且Unity无法修补它们。

**Windows Store Apps Plugin Settings**

**Windows商店App插件设置**

Options in the editor tab

编辑器的选项标签

| ***Property:*** | ***Function:*** |
| --- | --- |
| **SDK** | Limits plugin to Windows Store Apps built with *SDK 8.0* or *SDK 8.1*.  限制插件用于Windows商店App构建的SDK属于8.0还是8.1版本 |
| **CPU** | limit plugin to *32 bit*, *64 bit* or *ARM* players.  限制插件用于32位，64位还是ARM的播放器 |
| **Don’t process** | (Only applies for managed assemblies) Disables patching for this assembly. Patching is needed when assemblies contains classes serializable by Unity. In these cases, additional IL code is injected into the assemblies. If you know that assemblies doesn’t have such classes then it’s safe to disable the patching. **Note:** if the assembly isn’t patched and Unity tries to serialize at runtime, you’ll get an ‘Out of bounds’ error or something similar.  （仅应用在托管的程序集）禁止为此程序集打补丁。当程序集包含Unity可序列化的类时需要打补丁。在这些情况中，额外的IL代码被注入打程序集中。如果您知道程序集中没有这样的类，那它是安全的所以可以禁用修补。注意：如果程序集没有被修补并且Unity在运行时尝试序列化它，您将得到一个‘Out of bounds'的错误或者类似的错误。 |
| **Placeholder** | (Only applies for managed assemblies). With Windows Store Apps you can have plugins compiled against .NET Core, but because the Unity Editor runs on Mono, it will fail to recognize such assemblies. As a result, C# or JS files won’t be able to reference them. To overcome this, you have to provide an assembly compiled against .NET 3.5 with identical API which would act as a placeholder for the real plugin.  （仅应用在托管的程序集）针对Windows商店应用您可以使用针对.Net Core编译的插件，但是因为Unity编辑器在Mono上运行，所以它将无法识别此类程序集。因此，C#和JS文件无法引用他们，为了克服它，您必须提供一个使用相同API编译的.Net 3.5程序集，它将作为真正插件的placeholder。 |

**Placeholder Example**

For example, let’s say you have two assemblies:

例如，假设您有两个程序集

* **Plugins\WSA\MyPlugin.dll** - assembly compiled against .NET Core with Windows Runtime API inside.
* **Plugins\WSA\MyPlugin.dll** - 针对拥有Windows运行时API的.Net Core的程序集
* **Plugins\MyPlugin.dll** - assembly compiled against .NET 3.5 which has identical public API with dummy function implementations.
* **Plugins\MyPlugin.dll** - 针对具有虚函数实现的公有API的.Net 3.5版本的程序集

Click on **Plugins\WSA\MyPlugin.dll**, select Placeholder and pick **Plugins\MyPlugin.dll**.

点击**Plugins\WSA\MyPlugin.dll**，对Placeholder一项选择Plugins\MyPlugin.dll.

This way when Unity compile scripts it will reference **Plugins\MyPlugin.dll** file, but when Unity will copy plugins to final directory, it will copy **Plugins\WSA\MyPlugin.dll** instead of **Plugins\MyPlugin.dll**.

这种方式会在Unity编译脚本时引用**Plugins\MyPlugin.dll** 文件，但当Unity将插件复制到最终目录时，它将复制**Plugins\WSA\MyPlugin.dll**替换**Plugins\MyPlugin.dll**

**What happens if Unity won’t patch your assemblies?**

**如果Unity不为您的程序集做补丁时会发生什么？**

Unity injects serialization code into your assemblies, meaning if you have a class derived from MonoBehaviour in your plugin, and Unity doesn’t patch it, you might get a serialization error during runtime.

Unity注入序列化代码到您的程序集，意思是在您插件里继承MonoBehaivour的类，那么Unity不会修补它，您可能会在运行时得到一个序列化错误。

Both of them must share the same name. For example, you should place an Editor compatible plugin at Assets\Plugins\MyPlugin.dll, and a Windows Store Apps specific plugin at Assets\Plugins\WSA\MyPlugin.dll. When you’re working in Editor the Assets\Plugins\MyPlugin.dll will be used, and when you’re building to a Windows Store Apps app the Assets\Plugins\WSA\MyPlugin.dll will copied over to the build.

它们两者必须共享相同的名字。例如，您应该将编辑器的插件放在Assets\Plugins\MyPlugin.dll，并且将Windows商店应用的专有插件放在Assets\Plugins\WSA\MyPlugin.dll。当您在编辑器中工作时将使用Assets\Plugins\MyPlugin.dll，当您编译Windows商店应用程序时将会复制Assets\Plugins\WSA\MyPlugin.dll到构建里。