

Overview

SmartNS v2.0.1

SmartNS is a simple editor extension that monitors the creation of C# scripts and automatically adds a namespace to the scripts. The default behavior is for the namespace to mirror the physical path of the C# script within the unity project. For example, if you create a new script within a directory named 'Assets/Code/Enemies', SmartNS will add the declaration `namespace Assets.Code.Enemies {}` to the script, wrapping the content of the class.

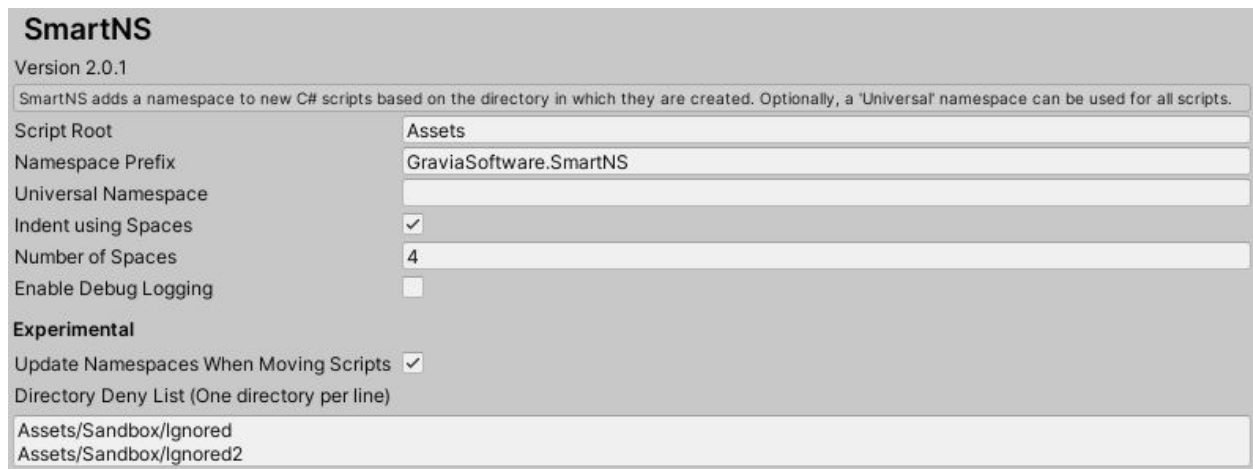
This behavior works whether you create a script via 'Create -> C# Script', or using other script creation methods, such as adding C# unit tests.

An optional experimental feature also the same namespace adjustment to be applied when moving scripts to a different directory, or when renaming a directory path containing scripts. Other options can be adjusted in the Project Settings window.

Setup

In Unity, simply import the package via 'Assets -> Import Package -> Custom Package...', or install the package from the Asset Store.

Options



The screenshot shows the 'SmartNS' settings window with the following configuration:

- Version 2.0.1**
- Description:** SmartNS adds a namespace to new C# scripts based on the directory in which they are created. Optionally, a 'Universal' namespace can be used for all scripts.
- Script Root:** Assets
- Namespace Prefix:** GraviaSoftware.SmartNS
- Universal Namespace:** (empty)
- Indent using Spaces:** ☒
- Number of Spaces:** 4
- Enable Debug Logging:** ☐
- Experimental**
 - Update Namespaces When Moving Scripts:** ☒
 - Directory Deny List (One directory per line):**
 - Assets/Sandbox/Ignored
 - Assets/Sandbox/Ignored2

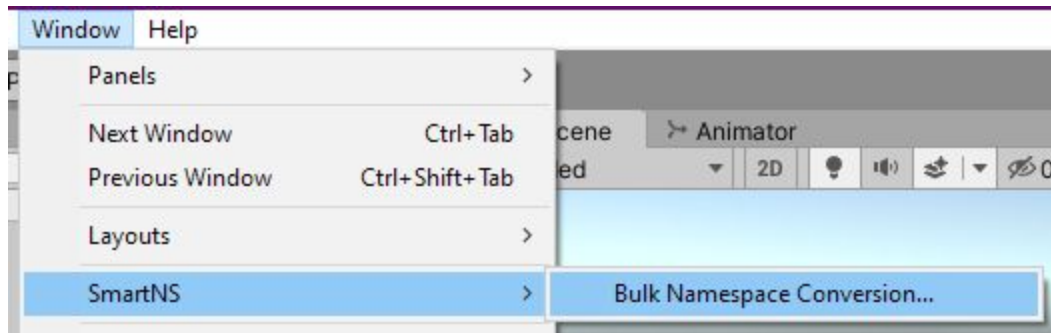
After installing SmartNS, you can modify the behavior by adjusting the Project Settings under 'Edit -> Project Settings...', and clicking on SmartNS. All SmartNS settings are project-specific

The options are:

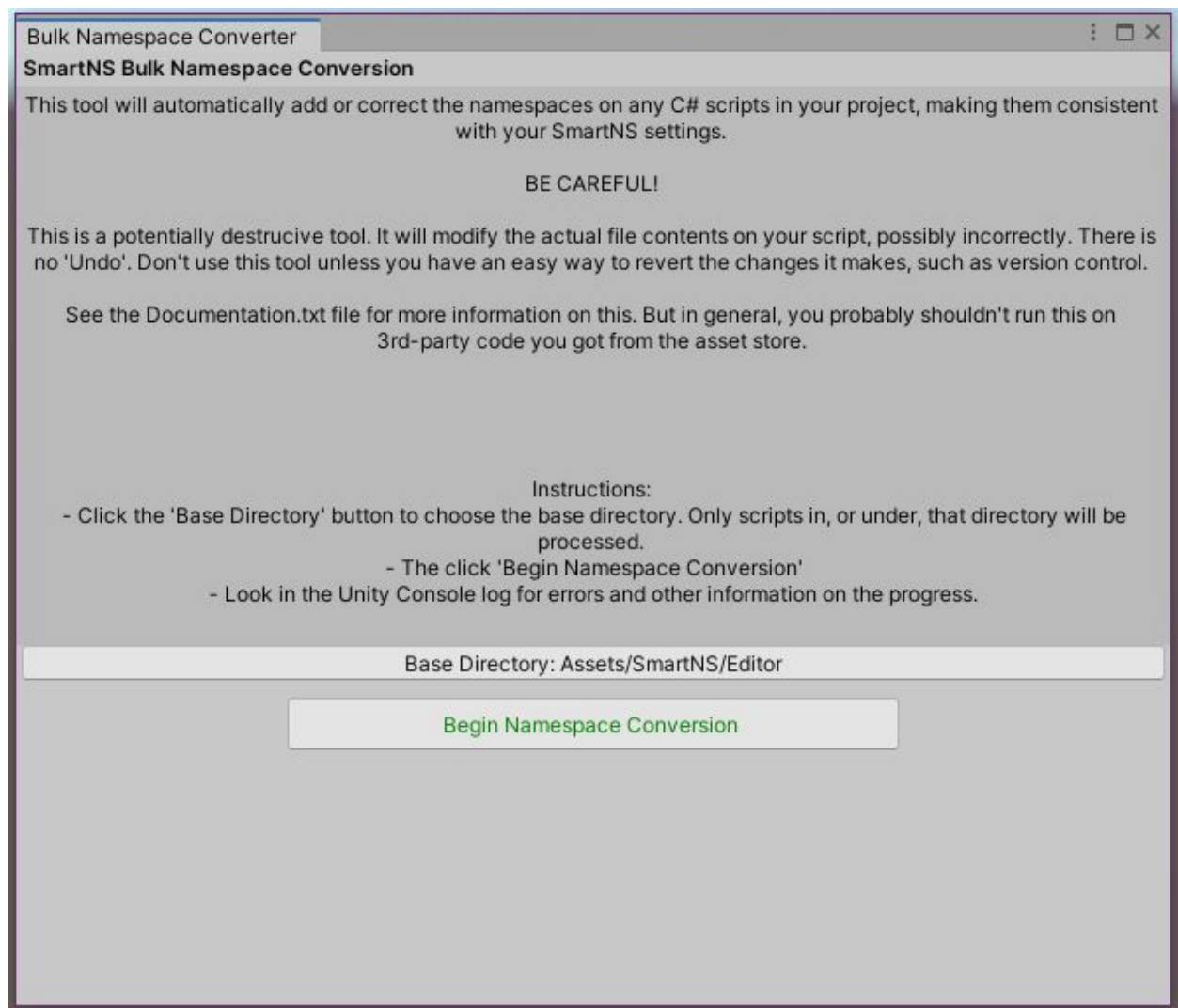
- **Script Root:** Whatever you enter here will be stripped from the namespace. Since many projects keep all of their assets under a folder named Assets, it's usually preferable for "Assets" not to be included in the namespaces. If you keep all scripts in a directory named "Assets/Code", you might want to set this *Script Root* value to "Assets.Code" to remove that from the namespaces.
- **Namespace Prefix:** This is a value that will be prepended to all dynamically created namespaces. For example, if you place a script under /Code/Controllers, the namespace would normally be "Code.Controllers". But if you set the *Namespace Prefix* to "MyCompany", the generated namespace will be "MyCompany.Code.Controllers". This helps to keep namespace fully-qualified between projects.
- **Universal Namespace:** This overrides the "smart" directory-based namespace generation, and instead uses the entered namespace in all cases. For example, if you set this to "MyUniversalNamespace", then every script created will use that namespace instead of a dynamically generated namespace based on its directory structure.
- **Indent using Spaces:** The default behavior is to wrap the `public class...` declaration in the namespace declaration, and to indent all lines between the opening and closing braces using tabs. If this is checked, spaces will be used instead.
- **Number of Spaces:** When using *Indent using Spaces*, this is the number of spaces that will be used.
- **Enable Debug Logging:** This writes some log information to the console when scripts are created.
- **Update Namespaces When Moving Scripts (Experimental):** Originally SmartNS only added namespaces to newly created scripts. This experimental feature extends that behavior to also update the namespaces of existing scripts. This allows your namespaces to stay in-synch as you move files around within your project, or rename a parent directory.
- **Directory Deny List (Experimental):** Your project may contain some directories where you'd prefer SmartNS didn't run. For example, if you keep 3rd party scripts in a directory "Asset/Code/ThirdParty", you might want to add that path to the Directory Deny List, to prevent SmartNS from acting on files in that path.

Bulk Namespace Conversion (Experimental)

If you're adding SmartNS to an existing project, you can use the Bulk Namespace Conversion tool to go through the entire project, adding or updating script namespaces as though you had originally created them when SmartNS was installed. You can find this feature under Window -> SmartNS -> Bulk Namespace Conversion...



This opens the Converter window. Heed the warning, and only use this tool if you can easily revert the changes by some other means, such as version control. This could break your project, and there's no built-in Undo:



Click on “Begin Namespace Conversion” to begin the process. It should be relatively fast, even for large projects, and the progress bar will show how it’s coming along.