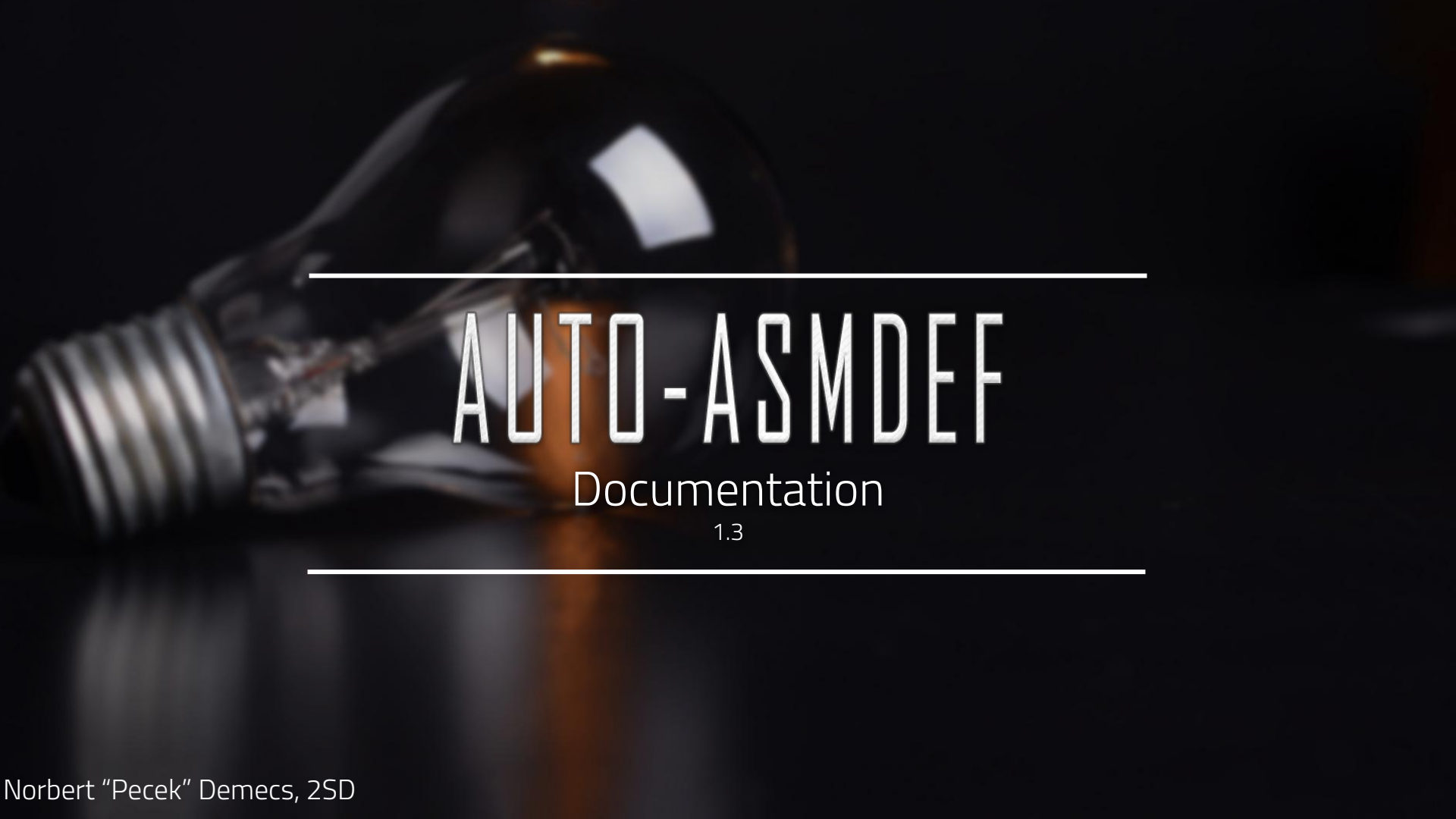


AUTO-ASMDEF

Documentation

1.3

Norbert “Pecek” Demecs, 2SD

Table of Contents

AUTO-ASMDEF

AUTO-ASMDEF

TL;DR;

1. Open up the Folder Hierarchy tab(or use the RegEx textbox if you are into that), deselect everything *but* your own script folder
2. Hit “Start Process”

That’s it!

1. The problem

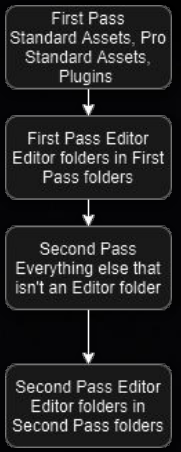
Compilation time depends on the size of your codebase, basically the more code you have the longer you have to wait between each iteration.

You'll inevitably end up in a situation where you tweak a single value in your code and then you have to wait up to a minute just to realize you made a typo and instead of .5 you wrote 5. You fix it, and then wait another minute to see that .5 was never going to work anyway. This makes iteration a nightmare.

AUTO-ASMDEF

2. Unity's first solution

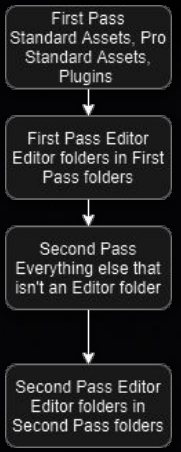
Unity introduced the Special Folders system, you can essentially break down your game into 4 pieces, each will be compiled after the other in a linear order. This speeds up compilation time because unless you change something in your first pass, you don't have to recompile it.



More information about the topic

2. The problem with Unity's first solution

This system in theory would be great. You can put everything you don't change regularly into your Plugins folder and your compilation time would be much shorter regardless of how much code you have in your First Pass.

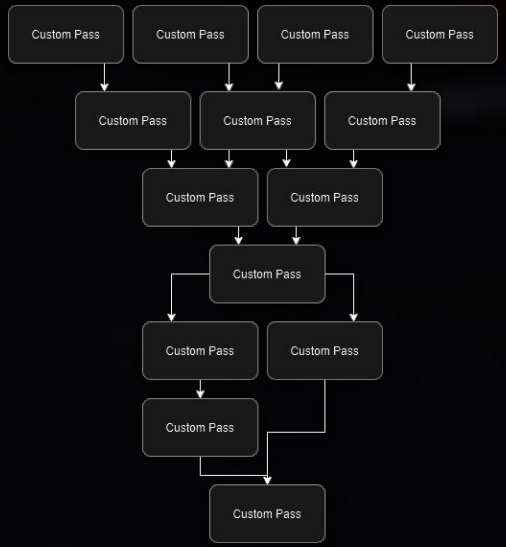


But this would require people to follow conventions - and they don't. So you end up with assets cluttering your project, making your compilation time skyrocket.

2. Unity's second solution

AUTO-ASMDEF

Unity introduced Assembly Definitions to solve this problem - and it's great, now you can have as many "Passes" as you need, and make them dependent only on code that's actually needed. This is a really flexible system, but it does have a major downside - it can take a really long time to set up even in a semi-complex project.



More information about the topic

AUTO-ASMDEF

3. Auto-ASMDEF

Auto-ASMDEF takes pain out of the equation, and provides the best of both worlds.

It'll analyze your entire project(or selected folders) and put them in its custom FirstPass, FirstPassEditor, SecondPass, or SecondPassEditor assemblies - where you can access them easily from outside if needed. Everything that was in Unity's First Pass will end up a custom assembly named FirstPass, and so on.

Nothing will break, nothing will be removed or moved.

AUTO-ASMDEF

Step-by-Step

Open up the editor window
Window-> 2SD-> Auto Assembly Definition Manager

Debug Settings - if the process fails you can turn these settings on to see when it gets stuck. You can also manually rescan the folders or run an automatic cleanup.

[Advanced] Always Referenced Assemblies - by default everything is referenced, this will be the best fit for almost every case, but in some

Folder Hierarchy - select everything other than your game's code

Exclude via RegEx - you can manually exclude folders with RegEX support

Start Process - it'll start the creation of ASMDEF and ASMREF files, and sets up the dependencies.

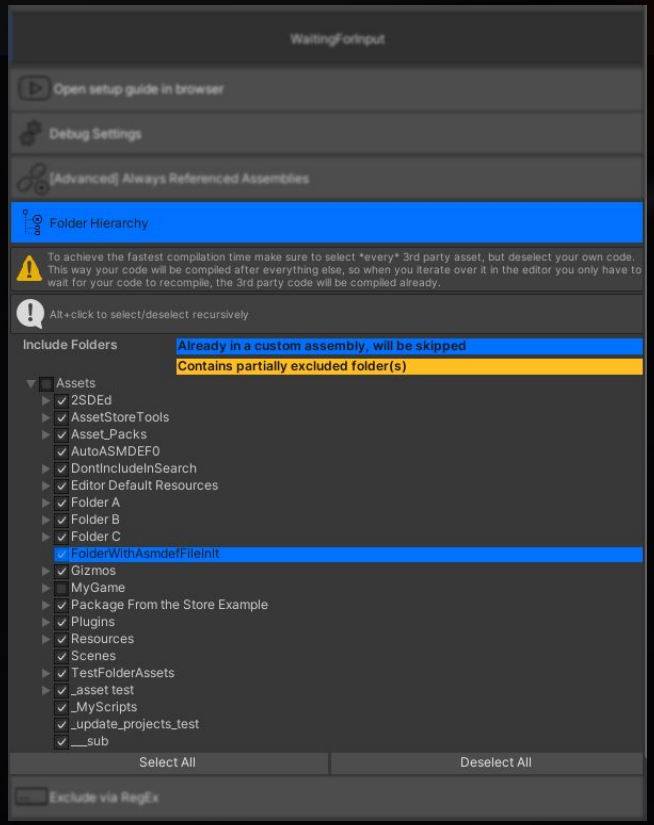
Step-by-Step

The Folder Hierarchy tab should be your first destination. Make sure you have selected each and every folder you want to include in your assemblies.

If a folder is already in a custom assembly then it'll be skipped no matter what, whether it's marked or not.

Realistically you would want to select everything but the folder you are actually working, this way you can put every third party script into a custom assembly and your code can compile much faster.
In the figure(on the right), the contents of "MyGame" will be compiled after everything else.

Tip: this is a default Unity TreeView, meaning you can use the shortcuts.
Alt+Click on the arrow will open\close children recursively
Alt+Click on a checkmark will enable\disable children recursively



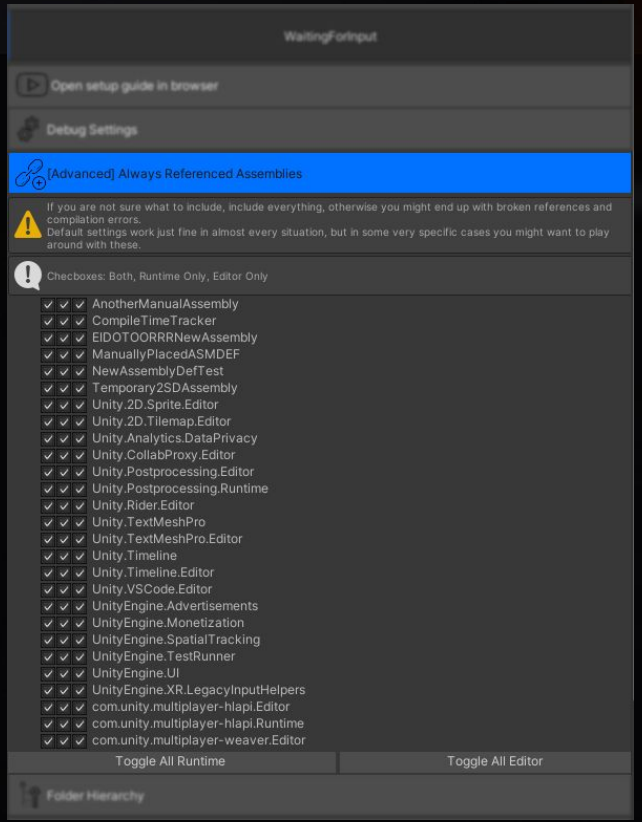
Step-by-Step

Now let's move to [Advanced] Always Referenced Assemblies tab.

TL;DR; - every assembly is referenced by default, and you most likely don't want to touch it.

The name "reference" isn't describing well what's happening - it's not a reference as you, a programmer would think so, it simply means the assemblies that are "referenced" will be compiled before this assembly, so if you want you can access them, but they won't be included with your build just because the assembly is "referenced". It really is just a way to tell Unity the order of compilation, nothing else.

There are some specific scenarios when you would want to change these settings(for example if you want to use Unity's UnitTest) so the option is here, but other than that you wouldn't even have to open this tab.



AUTO-ASMDEF

Step-by-Step

Now press “Start Process”, and the tool is going to create the required *.asmdef, and *.asmref files throughout your project.

For asmdef files a new folder will be automatically created in the root Assets folder(Assets/AutoASMDEFn).

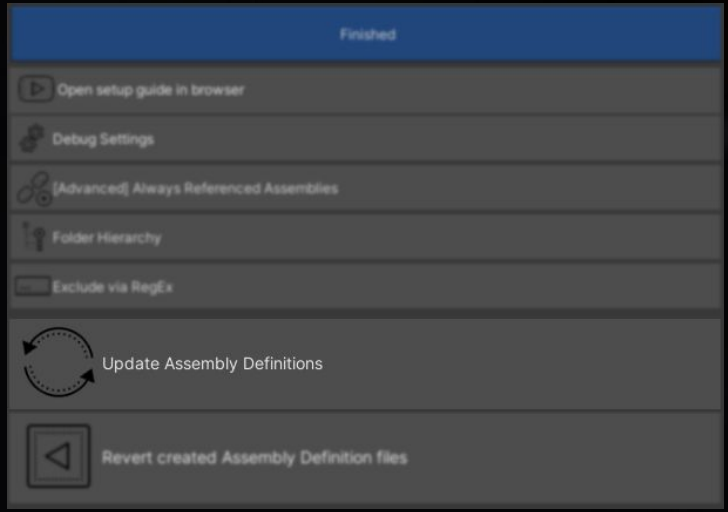
Note: the first compilation is going to take longer than usual, as you move pretty much everything, but after that as long as you only change the contents of the excluded folder(s) you only ever have to recompile those.

Step-by-Step

Now what if we add new scripts to folders that didn't exist when we first run the tool?

Instead of "Start Process", you will find two new buttons in the Editor Window.

"Update Assembly Definitions", as the name suggests is used for updating the existing asmdef files. This comes super handy when using source control.



Step-by-Step

Now what if we want to reverse the process?
Just hit the “Revert created Assembly Definition files” button, and wait for Unity to recompile.

That's it!

A hot-tempered person stirs up conflict, but the one who is patient calms a quarrel.
Proverbs 15:18