Include-What-You-Use (**IWYU**), as the name implies, means that the Engine's source code only includes the dependencies that it needs to compile. The purpose of IWYU is to avoid including monolithic header files, such as `Engine.h` or `UnrealEd.h`, thereby mitigating superfluous dependencies. The following reference guide tells you what it means to IWYU, including a high-level explanation of how to enable IWYU, ensuring that your project adheres to IWYU conventions. Additionally, if you opt into using IWYU mode for your game project(s), you will learn some general tips that will help you get the most out of working in IWYU mode.

NOTE

IWYU mode is disabled by default for games and game plugins; however, IWYU mode is enabled by default for the Engine and Engine plugins.

What it Means to IWYU

In previous versions of Unreal Engine 4 (UE4), the majority of engine functionality was included via large, module-centric header files, such as `Engine.h` and `UnrealEd.h` and fast compile times were dependent on those files being compiled quickly through Precompiled Header (PCH) files. As the engine grew, this became a bottleneck.

With IWYU, every file includes only what it needs, and any PCH file we choose to use, purely acts as a layer of optimization on top of the underlying source files. They can be modified to minimize build times, independently of changing the source files themselves, and without affecting whether the code compiles successfully or not.

When writing IWYU code, there are four specific conventions that we adopt:

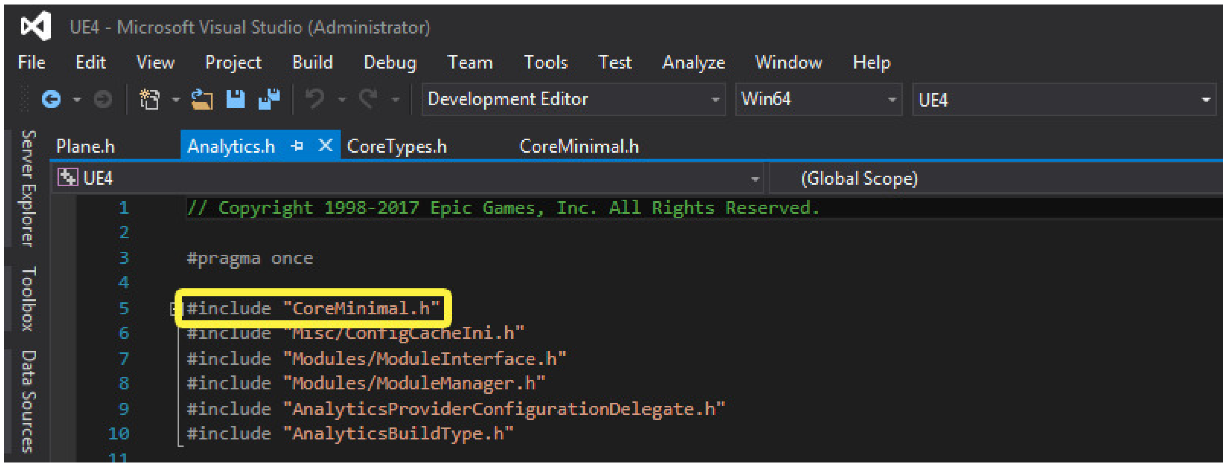
1. **All header files include their required dependencies.**
2. **.cpp files include their matching *.h files first.**
3. **PCH files are no longer explicitly included.**
4. **Monolithic header files are no longer included.**

IWYU Conventions

The following descriptions of IWYU conventions should give you a good idea about what it means to IWYU.

1. **All header files include their required dependencies.**

   - There is a **CoreMinimal** header file containing a set of ubiquitous types (including FString, FName, TArray, etc.) from UE4's Core programming environment.
   - The `CoreMinimal` header file (located under the UE4 root directory at `\Engine\Source\Runtime\Core\Public\CoreMinimal.h`) is included first by most of the Engine's header files.



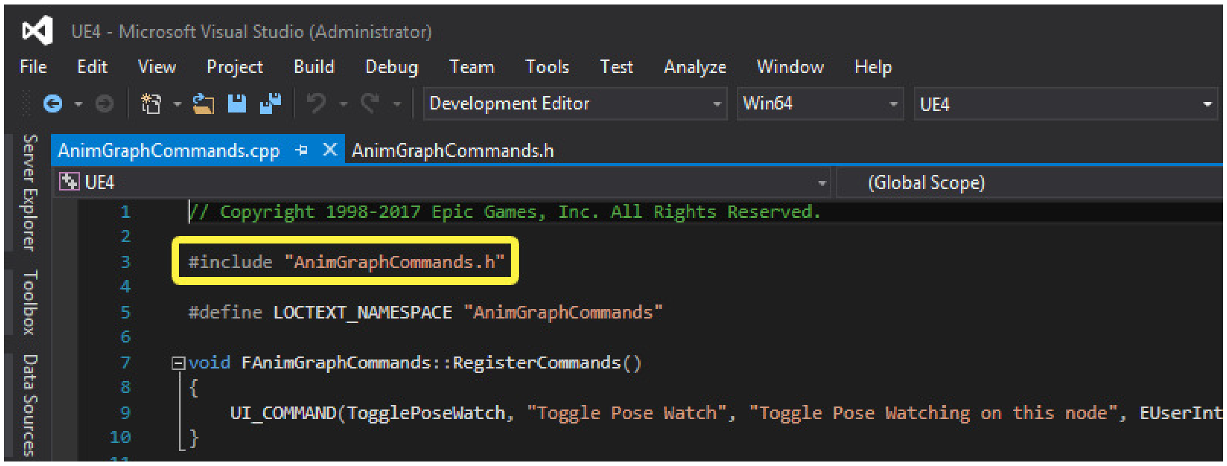
```
// Copyright 1998-2017 Epic Games, Inc. All Rights Reserved.

#pragma once

#include "CoreMinimal.h"
#include "Misc/ConfigCacheIni.h"
#include "Modules/ModuleInterface.h"
#include "Modules/ModuleManager.h"
#include "AnalyticsProviderConfigurationDelegate.h"
#include "AnalyticsBuildType.h"
```

- Within the `Core` module, most header files include the `CoreTypes.h` header file first. This only includes typedefs for primitive C++ types, UE4 build macros, and directives to configure the compile environment.

TIP

The main takeaway is that every header file now includes everything that it needs to compile.

2. **.cpp files include their matching *.h file first.**



```
// Copyright 1998-2017 Epic Games, Inc. All Rights Reserved.

#include "AnimGraphCommands.h"

#define LOCTEXT_NAMESPACE "AnimGraphCommands"

void FAnimGraphCommands::RegisterCommands()
{
    UI_COMMAND(TogglePoseWatch, "Toggle Pose Watch", "Toggle Pose Watching on this node", EUserInt
}
```

- To verify that all of your source files include all of their required dependencies, compile your game project in non-unity mode with PCH files disabled.

3. **PCH files are no longer explicitly included.**

- Although PCH files are still used, they're force-included on the compiler command line by UnrealBuildTool (UBT).

4. **Monolithic header files are no longer included.**

- The compiler will emit a warning if your Engine code includes monolithic header files (such as `Engine.h` or `UnrealEd.h`).

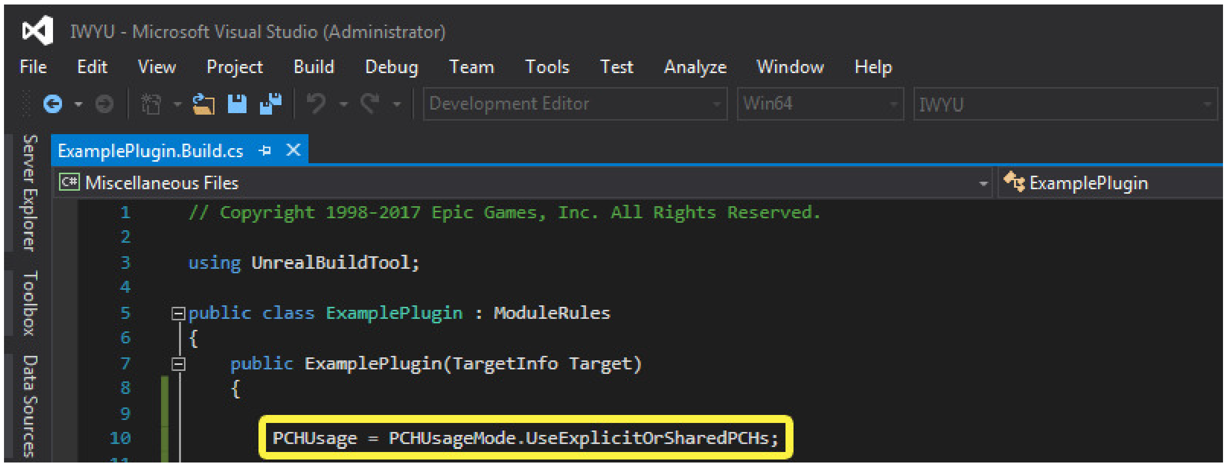
NOTE

Monolithic header files still exist in UE4 for compatibility with game projects, and (by default) a warning won't be emitted if your game projects include them.

Verifying IWYU is Enabled

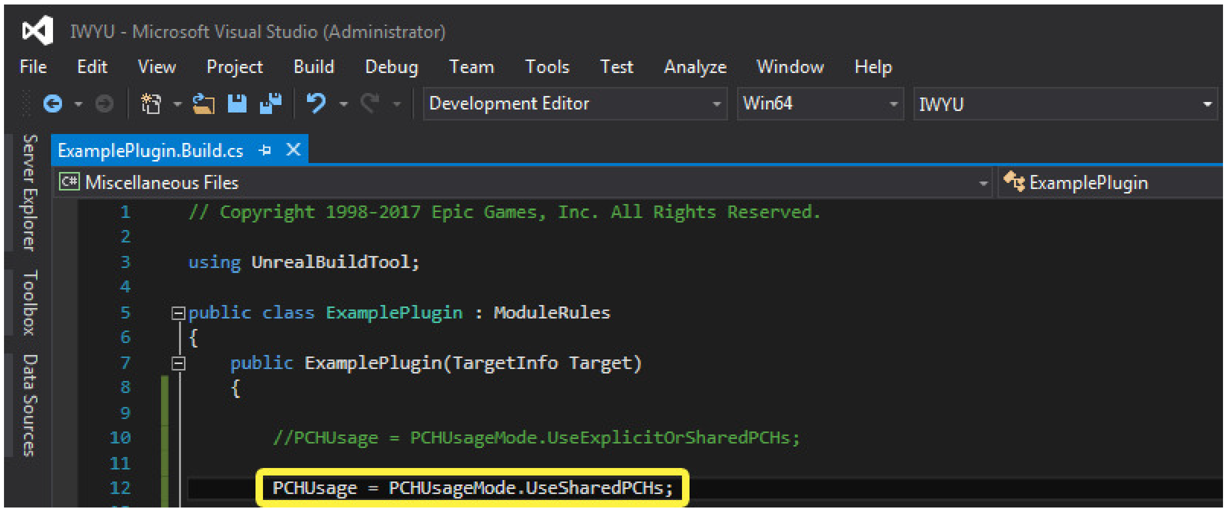
Before establishing IWYU conventions with the release of version 4.15, UE4 code typically included a PCH file at the top of every CPP file, which is contrary to what a developer wants their IWYU compliant code to include. Following IWYU conventions, PCH files can be thought of as layers of compile-time optizations that are applied separately from how the code was originally authored. So, rather than composing and including PCH files, we leave it to UBT to decide which PCH file to use (if any).

If you want to verify that IWYU is enabled, ensuring that a module complies with IWYU conventions, open the module's *.build.cs file and verify that `PCHUsage` is set to `PCHUsageMode.UseExplicitOrSharedPCHs`.


IWYU - Microsoft Visual Studio (Administrator)
File Edit View Project Build Debug Team Tools Test Analyze Window Help
Development Editor
Win64
IWYU
Server Explorer Toolbox Data Sources
ExamplePlugin.Build.cs
Miscellaneous Files
ExamplePlugin

```
// Copyright 1998-2017 Epic Games, Inc. All Rights Reserved.

using UnrealBuildTool;

public class ExamplePlugin : ModuleRules
{
    public ExamplePlugin(TargetInfo Target)
    {

        PCHUsage = PCHUsageMode.UseExplicitOrSharedPCHs;
```

Setting `PCHUsage` to `PCHUsageMode.UseExplicitOrSharedPCHs` creates an explicit PCH file for a module only if it has a `PrivatePCHHeaderFile` setting in the module's *.build.cs file. Otherwise, the module will share a PCH with another module, saving the tool from generating more PCH files than necessary. Also, keep in mind that when you enable `UseExplicitOrSharedPCHs` mode, the source file must include its matching header file. Alternatively, if you want a module to opt-out of complying with IWYU conventions, you can set `PCHUsage` to `PCHUsageMode.UseSharedPCHs`.


IWYU - Microsoft Visual Studio (Administrator)
File Edit View Project Build Debug Team Tools Test Analyze Window Help
Development Editor
Win64
IWYU
Server Explorer Toolbox Data Sources
ExamplePlugin.Build.cs
Miscellaneous Files
ExamplePlugin

```
// Copyright 1998-2017 Epic Games, Inc. All Rights Reserved.

using UnrealBuildTool;

public class ExamplePlugin : ModuleRules
{
    public ExamplePlugin(TargetInfo Target)
    {

        //PCHUsage = PCHUsageMode.UseExplicitOrSharedPCHs;

        PCHUsage = PCHUsageMode.UseSharedPCHs;
```

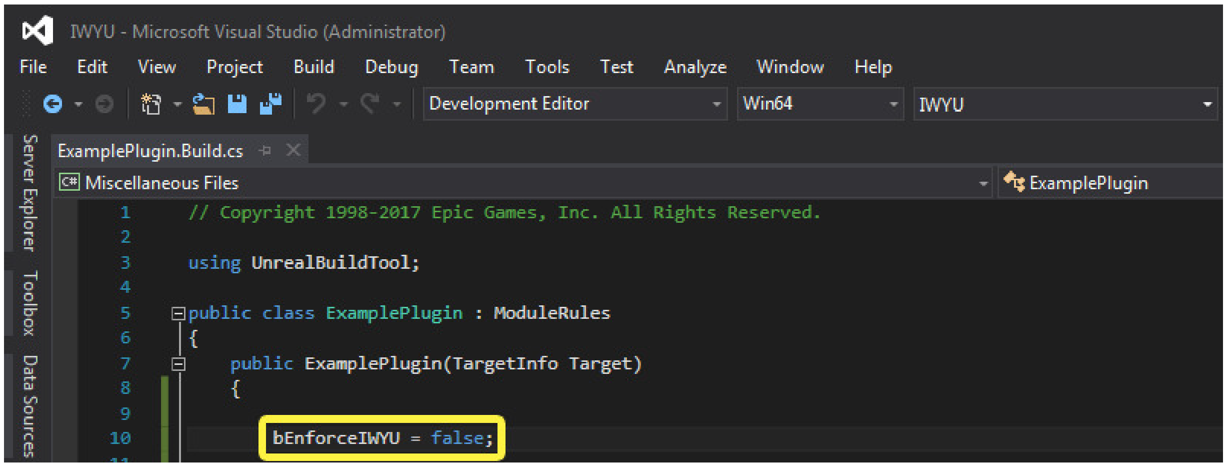
NOTE

After converting the engine's code base to an IWYU model, we've observed significant improvements with UE4 compile times.

Running in IWYU Mode

If you're running your game in IWYU mode, you'll need to ensure that your *.cpp file(s) include their corresponding* .h file(s) first. This is a useful practice, because it enables the compiler to ensure that the *.h file includes all of its required dependencies (when PCH files and unity builds are disabled). Unreal Build Tool (UBT) will emit a warning if you don't include the matching header file first (for its corresponding CPP file).

If you want to disable the compiler from emitting warnings, you can set `bEnforceIWYU` to `false` in the module's *.build.cs file.


IWYU - Microsoft Visual Studio (Administrator)
File Edit View Project Build Debug Team Tools Test Analyze Window Help
Development Editor
Win64
IWYU
ExamplePlugin.Build.cs
Miscellaneous Files
ExamplePlugin
Server Explorer
Toolbox
Data Sources

```
// Copyright 1998-2017 Epic Games, Inc. All Rights Reserved.

using UnrealBuildTool;

public class ExamplePlugin : ModuleRules
{
    public ExamplePlugin(TargetInfo Target)
    {

        bEnforceIWYU = false;
```

TIP

If you want to disable the warning for an entire target, you can set `bEnforceIWYU` to false in the *.target.cs file.

General Tips

If you want your game to opt-in to IWYU, there are a few tips to keep in mind:

1. Include `CoreMinimal.h` at the top of each header file.
2. To verify that all of your source files include all of their required dependencies, compile your game project in non-unity mode with PCH files disabled.
3. If you need to access **UEngine** or **GEngine**, which are defined in `Runtime\Engine\Classes\Engine\Engine.h`, you can `#include Engine/Engine.h` (distinguishing from the monolithic header file, which is located at `Runtime\Engine\Public\Engine.h`).
4. If you use a class that the compiler doesn't recognize, and don't know what you need to include may be missing the header file. This is especially the case if you are converting from non-IWYU code that compiled correctly. You can look up the class in the API Documentation, and find the necessary modules and header files at the bottom of the page.

Additional Resource

In an effort to help our users convert existing C++ projects into an IWYU style, we released **IncludeTool**, which you can find in `[UE4Root]\Engine\Source\Programs\IncludeTool`.