

JsonAssets Plugin

The JsonAssets plugin supports Windows, Mac, Linux, Android, and iOS. Since it is not compatible with 4.19 or below a **minimum engine version of 4.20 is required** to install.



DOWNLOAD

This plugin can be downloaded from GitHub at the following address:

<https://github.com/tracerinteractive/UnrealEngine/releases>

4.26 Latest release

4.26.0 172ab38

tracerinteractive released this 9 hours ago

Assets 6

HttpLibrary-4.26.zip	29.5 MB
JsonAssets-4.26.zip	84.2 MB
JsonLibrary-4.26.zip	49 MB
WebUI-4.26.zip	83.1 MB

Each version is also compatible with all minor engine updates which means the 4.26.0 version of the plugin will work with any corresponding hotfix such as 4.26.1 or 4.26.2 as well.

You must have a GitHub account linked to your Epic Games account!

Setup Instructions: <https://www.unrealengine.com/ue4-on-github>



Otherwise you will receive the previous 404 error if you are not signed in with a linked account.

TRY + BUY

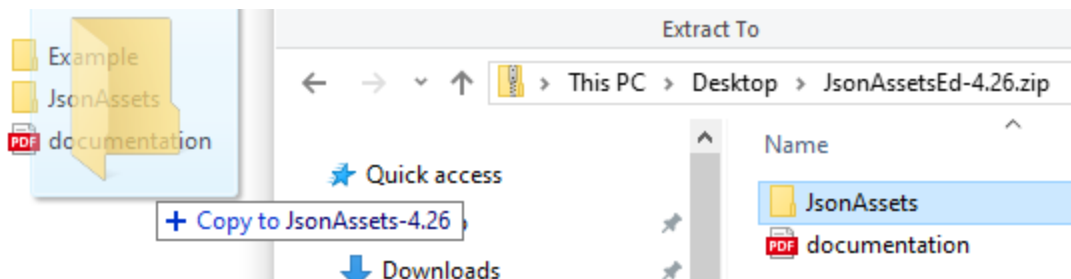
If you would like to try the JsonAssets plugin then **the following change is required** in the `JsonAssets.uplugin` file before installing. Otherwise if you downloaded the plugin from our website keep reading without making this edit. To try the JsonAssets plugin without the “editor” part, remove this section:

```
    },  
    {  
      "Name": "JsonAssetsEd",  
      "Type": "Editor",  
      "LoadingPhase": "PostEngineInit",  
      "WhitelistPlatforms": [  
        "Win64",  
        "Mac",  
        "Linux"  
      ]  
    }  
  ],  
  "Plugins": [  
    ]
```

Don't forget to remove the leading comma before the opening brace! The entire block has been highlighted for you in the previous image. Once you've deleted this block you should save the file and skip to the install section in this document.

The “editor” part of the plugin is not included on GitHub. It is not required for existing projects that already contain Structure Instance assets to function nor is it necessary when compiling a public distribution of your project. You can still utilize UTXT assets without the “editor” part but they only show up in the content browser if you load an existing asset that references them.

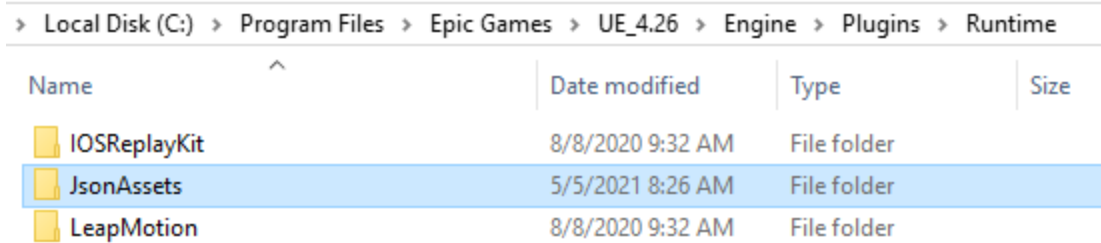
If you [downloaded the plugin from our website](#) then do not edit the `JsonAssets.uplugin` file from GitHub; instead copy/paste the “editor” files in `JsonAssetsEd.zip` to merge them with the files in the `JsonAssets.zip` from GitHub before installing:



This is as simple as dragging and dropping the `JsonAssets` folder from the “paid” zip (the one without the example project) to the “free” zip (the one with the example project).

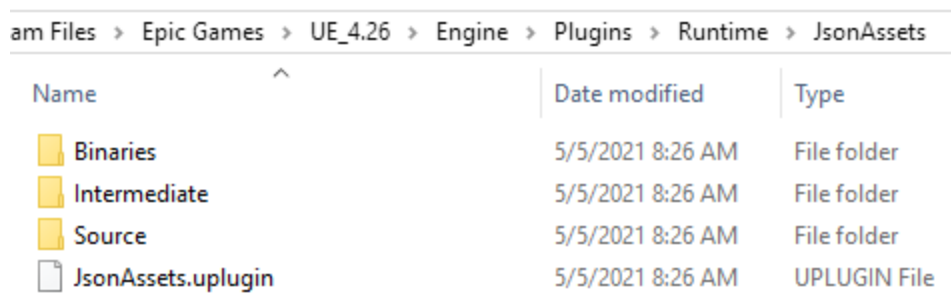
INSTALL

To install the JsonAssets plugin extract the downloaded files to the following engine folder:



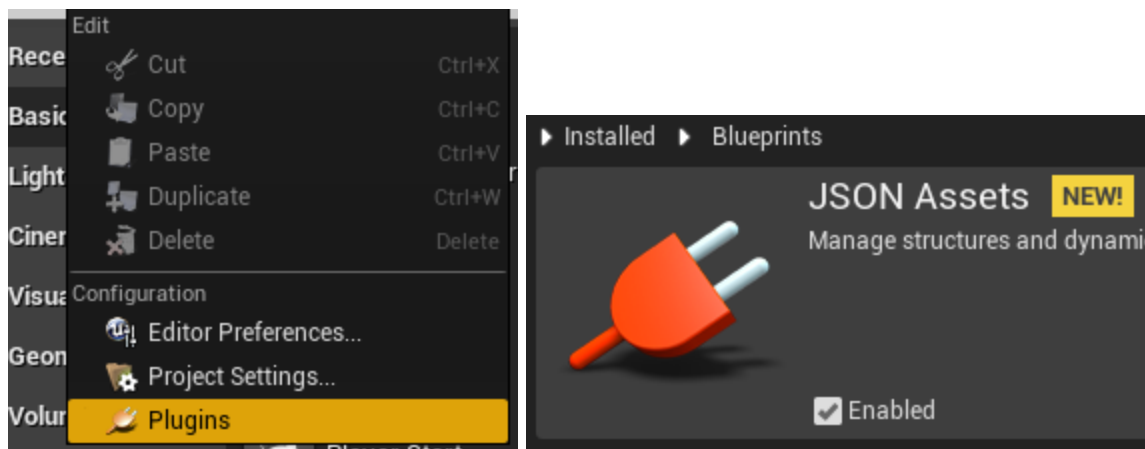
Name	Date modified	Type	Size
IOSReplayKit	8/8/2020 9:32 AM	File folder	
JsonAssets	5/5/2021 8:26 AM	File folder	
LeapMotion	8/8/2020 9:32 AM	File folder	

Also take note of the **UE_4.26** directory in the screenshots. You need to change this folder to the version that corresponds to the plugin version that was downloaded. *If you did not install your engine to the default directory then navigate to your custom installation folder instead.*



Name	Date modified	Type
Binaries	5/5/2021 8:26 AM	File folder
Intermediate	5/5/2021 8:26 AM	File folder
Source	5/5/2021 8:26 AM	File folder
JsonAssets.uplugin	5/5/2021 8:26 AM	UPLUGIN File

Then open your project and go to the “Plugins” option in the edit drop-down. Click on the “Blueprints” category and enable the JsonAssets plugin if it is not already enabled.



You have now successfully installed the JsonAssets plugin. Restart the editor to continue.

TABLE OF CONTENTS

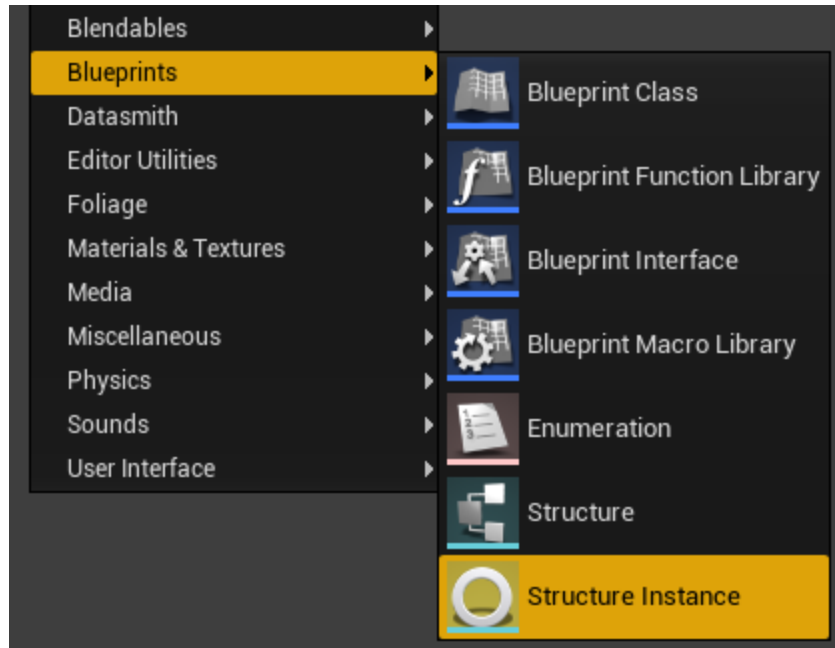
ASSETS	5
MENU	7
COOKING	8
SAVING	12
COMPILE	13

EPIC GAMES COMPLETELY BROKE .UTXT SAVING IN VERSION 4.25

(GO TO "SAVING" SECTION FOR DETAILS AND BUG FIX)

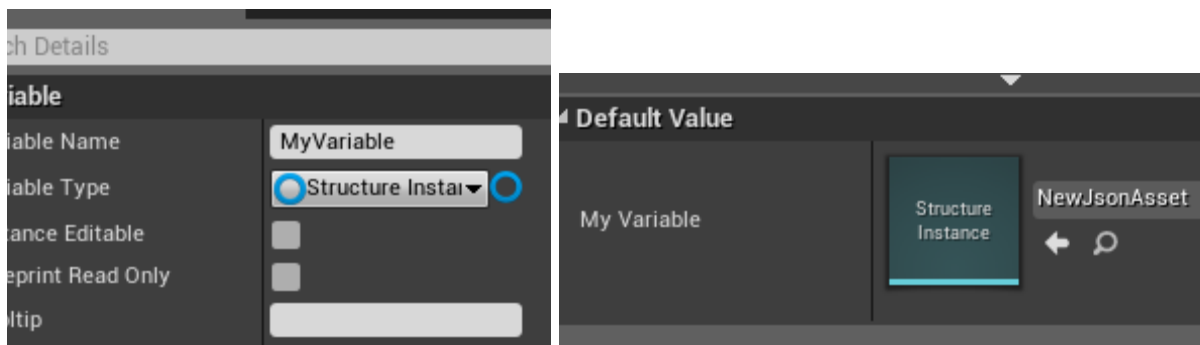
ASSETS

If you want to get started it's pretty simple. Just right click in the "content browser" and go to the "blueprints" section to create a "structure instance" asset.

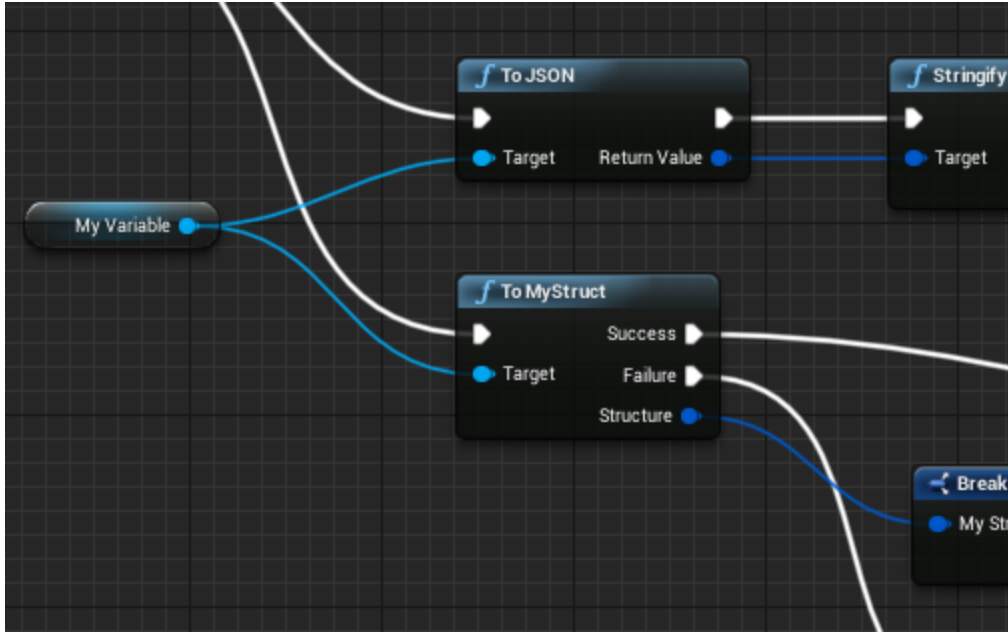


It will also ask you to pick a parent structure for your asset. This must be a blueprint-created structure (for now, this will be updated to support C++ structures in-editor in a future update).

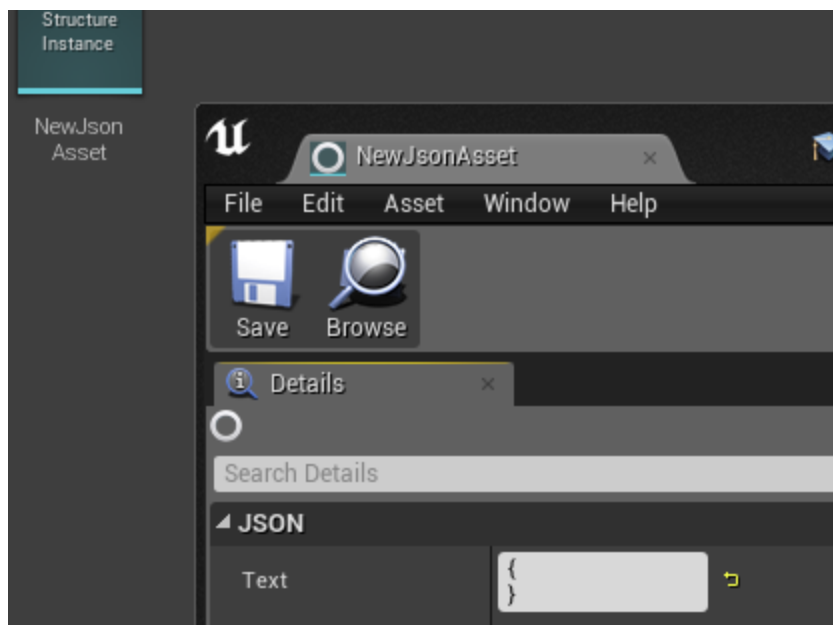
Then you can go into any blueprint and add a variable of type "structure instance" to your blueprint. Don't forget to set the asset reference in the default value!



Now you can access the data in your blueprint from the JSON asset using this variable along with the *ToJson* or *ToStructure* nodes.

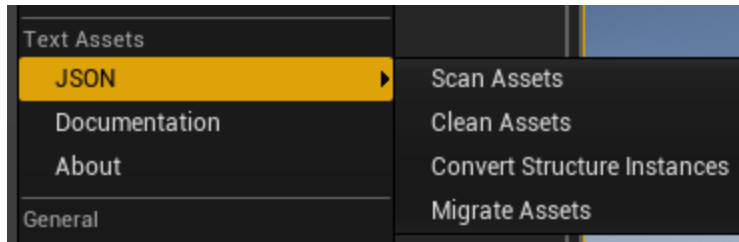


NOTE: No type data is actually needed for “structure instance” assets as they can technically represent any number of structures. Therefore the special \$type key is only used for opening the asset in the editor to make organized changes. You can always just open a UTXT “structure instance” file directly to edit raw JSON and then convert this into any structure in your blueprints. In fact if you save a “structure instance” asset as UTXT and then remove the \$type key after closing the editor, you’ll find that opening this asset next time you’re in-editor allows the editing of raw JSON since no type data is available.



MENU

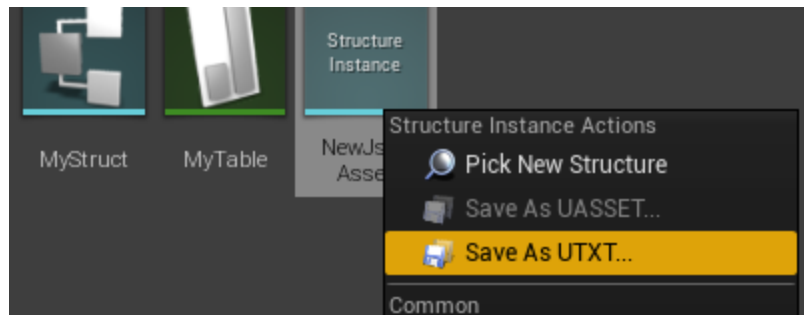
There's also a helpers menu available in the level editor tab. Just go to "*Window > JSON*" for a link to the most recent documentation along with the following menu options:



Scan Assets - This will find any new UTXT files in the *Content* folder and add them to the "content browser" if they aren't visible. You must refresh or change folders after executing this command (for now until updated, keeping the current folder open shows no changes).

Clean Assets - This will find any duplicate UASSET and UTXT files of the same name and delete whichever one is the older file of the two. *This might happen if a file was not properly closed which prevented the old asset format from being deleted after the new one was saved.*

Convert Structure Instances - This will find all "structure instance" assets in the *Content* folder that are in binary UASSET format and resave them all as text-based UTXT files instead. *You can also execute this action for any specific asset or a selection of assets in the "content browser" by right-clicking on any "structure instance" asset.*



Migrate Assets - This will find all UTXT files in the *Content* folder and resave them as binary UASSET files instead. *You should do this for all UTXT assets before migrating to new engine versions since this is an experimental feature in-engine and the UTXT format can change.*

COOKING

If you try to cook your project with UTXT assets you'll run into the following warnings:

```
LogInit: Display:
LogInit: Display: Warning/Error Summary (Unique only)
LogInit: Display: -----
LogInit: Display: LogSavePackage: Warning: /Game/MySample/MyHUD imported Serialize:/Game/MySample/MyJSON.MyJSON, but it was never saved as an export.
LogInit: Display: LogSavePackage: Warning: /Game/MySample/MyHUD imported Serialize:/Game/MySample/MyJSON, but it was never saved as an export.
LogInit: Display:
LogInit: Display: Success - 0 error(s), 2 warning(s)
LogInit: Display:
```

This is because UTXT assets are still considered an experimental feature and the engine developers never bothered to finish the UTXT implementation (which is just like most things they never finish). Therefore these assets will not work in cooked builds, period. The only way to cook your project would be to utilize the “migrate assets” option under “*Window > JSON*” to resave all UTXT files as UASSET files instead.

However even though the engine developers didn't bother to build out this feature end-to-end, we decided to do it for you. That's why a precompiled **UE4Editor-UnrealEd.dll** file is included in your download. All you have to do is replace this file in your engine installation:

Name	Date modified	Type	Size
UE4Editor-UndoHistory.dll	4/26/2021 8:11 AM	Application extens	818 KB
UE4Editor-UnrealAudio.dll	4/26/2021 8:11 AM	Application extens	360 KB
UE4Editor-UnrealEd.dll	5/10/2021 1:35 AM	Application extens	25,705 KB
UE4Editor-UnrealEd_.dll	4/26/2021 8:11 AM	Application extens	24,338 KB
UE4Editor-UnrealEdMessages.dll	4/26/2021 8:11 AM	Application extens	117 KB
UE4Editor-VectorVM.dll	4/26/2021 8:11 AM	Application extens	394 KB

Just navigate to the Engine/Binaries/Win64/ directory in your engine installation folder and rename **UE4Editor-UnrealEd.dll** with an underscore as shown in the screenshot. Then copy/paste the replacement file into this folder. Now your experimental UTXT assets will cook into UASSET files automatically. **DO NOT REPLACE THIS DLL FILE IF YOU DOWNLOADED UE4 FROM GITHUB; IT'S ONLY FOR ENGINE INSTALLATIONS VIA THE EPIC LAUNCHER!**

So this hack is **required to cook text-based UTXT assets**, otherwise you will be forced to save “structure instance” assets in binary UASSET format before cooking. *If you compiled the engine yourself from source (such as downloading the code from GitHub) then implement the following changes to manually fix this issue...*

ONLY CONTINUE READING IF YOU ARE NOT REPLACING THE DLL! Start by opening the `/Engine/Source/Editor/UnrealEd/Private/Cooker/PackageNameCache.h` file and search for the `FPackageNameCache::DoesPackageExist(...)` function around line 100:

```
bool FPackageNameCache::DoesPackageExist(const FName& PackageName, FString* OutFilename) const
{
    if (!AssetRegistry)
    {
        return FPackageName::DoesPackageExist(PackageName.ToString(), NULL, OutFilename, true);
    }

    TArray<FAssetData> Assets;
    AssetRegistry->GetAssetsByPackageName(PackageName, Assets, /*bIncludeOnlyDiskAssets*/ true);

    if (Assets.Num() <= 0)
    {
        return FPackageName::DoesPackageExist(PackageName.ToString(), NULL, OutFilename, true);
    }
}
```

Just change the `false` boolean to `true` in the `FPackageName::DoesPackageExist(...)` function and then copy/paste this exact same line to the `return false;` line as well. This will only apply to versions 4.25 and below. If you are editing version 4.26 or higher then make these changes instead:

```
if (!AssetRegistry)
{
    return FPackageName::DoesPackageExist(PackageNameStr, NULL, OutFilename, true);
}

TArray<FAssetData> Assets;
AssetRegistry->GetAssetsByPackageName(PackageName, Assets, /*bIncludeOnlyDiskAssets =*/true);

if (Assets.Num() <= 0)
{
    return FPackageName::DoesPackageExist(PackageNameStr, NULL, OutFilename, true);
}

if (OutFilename)
{
    const bool ContainsMap = Algo::FindByPredicate(Assets, [](const FAssetData& Asset) { return Asset
```

Find the `/Engine/Source/Editor/UnrealEd/Private/Commandlets/PackageUtilities.cpp` file after that and search for the `NormalizePackageNames(...)` method around line 100:

```
// now apply any filters to the list of packages
for ( int32 PackageIndex = PackagePathNames.Num() - 1; PackageIndex >= 0; PackageIndex-- )
{
    FString PackageExtension = FPaths::GetExtension(PackagePathNames[PackageIndex], true);
    if ( !FPackageName::IsPackageExtension(*PackageExtension) && !( FPackageName::GetTextAssetPackageExtension() == PackageExtension
        || FPackageName::GetTextMapPackageExtension() == PackageExtension ) )
    {
        // not a valid package file - remove it
        PackagePathNames.RemoveAt(PackageIndex);
    }
}
```

Don't forget the exclamation before the first parenthesis! This will only apply to versions 4.24 and below. If you are editing version 4.25 or higher then make this change instead:

```
// now apply any filters to the list of packages
for ( int32 PackageIndex = PackagePathNames.Num() - 1; PackageIndex >= 0; PackageIndex-- )
{
    FString PackageExtension = FPaths::GetExtension(PackagePathNames[PackageIndex], true);
    if ( !FPackageName::IsPackageExtension(*PackageExtension) && !FPackageName::IsTextPackageExtension(*PackageExtension) )
    {
        // not a valid package file - remove it
        PackagePathNames.RemoveAt(PackageIndex);
    }
}
```

Now find the `/Engine/Source/Editor/UnrealEd/Private/CookOnTheFlyServer.cpp` file and search for the `UCookOnTheFlyServer::SaveCookedPackages(...)` method around line 4500:

```
for (int32 PlatformIndex = 0; PlatformIndex < TargetPlatforms.Num(); ++PlatformIndex)
{
    SavePackageResults.Add(FSavePackageResultStruct(ESavePackageResult::Success));
    const ITargetPlatform* Target = TargetPlatforms[PlatformIndex];
    FString PlatFilename = Filename.Replace(TEXT("[Platform]"), *Target->PlatformName())
    .Replace(*FPackageName::GetTextAssetPackageExtension(), *FPackageName::GetAssetPackageExtension());

    FSavePackageResultStruct& Result = SavePackageResults[PlatformIndex];
```

You must add a second `Replace(...)` to the `PlatFilename` variable as shown. Don't forget to remove the semicolon from the previous line! *You don't technically have to put this on a new line either, but we've done so in this example to make it easier to screenshot.*

Then scroll down to the `UCookOnTheFlyServer::CollectFilesToCook(...)` method around line 5300 (and remember that line numbers won't match up between different engine versions):

```
// get the dlc and make sure we cook that directory
FString DLCPath = FPaths::Combine(*GetBaseDirectoryForDLC(), TEXT("Content"));

TArray<FString> Files;
IFileManager::Get().FindFilesRecursive(Files, *DLCPath, *(FString(TEXT("")) + FPackageName::GetAssetPackageExtension()), true,
IFileManager::Get().FindFilesRecursive(Files, *DLCPath, *(FString(TEXT("")) + FPackageName::GetMapPackageExtension()), true,
IFileManager::Get().FindFilesRecursive(Files, *DLCPath, *(FString(TEXT("")) + FPackageName::GetTextAssetPackageExtension()), true,
IFileManager::Get().FindFilesRecursive(Files, *DLCPath, *(FString(TEXT("")) + FPackageName::GetTextMapPackageExtension()), true);
for (int32 Index = 0; Index < Files.Num(); Index++)
{
    FString StdFile = Files[Index];
```

Some of the code is cutoff in these screenshots. However everything not shown in the image matches the previous lines with only the `FPackageName` function being different each time, so these updates are easy if you copy/paste the preceding line. *The previous image however is only applicable for versions 4.25 and below. It no longer exists in versions 4.26 or higher, but the next two images will be necessary...*

Now scroll down about 20 lines and make the same type of change, and finally one more time about another 20 lines below that:

```

TArray<FString> Files;
IFileManager::Get().FindFilesRecursive(Files, "CurrEntry, *(FString(TEXT(""))) + FPackageName::GetAssetPackageExtension()), tr
IFileManager::Get().FindFilesRecursive(Files, "CurrEntry, *(FString(TEXT(""))) + FPackageName::GetTextAssetPackageExtension())
for (int32 Index = 0; Index < Files.Num(); Index++)
{
    // If no packages were explicitly added by command line or game callback, add all
    if (FilesInPath.Num() == InitialPackages.Num() || bCookAll)
    {
        TArray<FString> Tokens;
        Tokens.Empty(2);
        Tokens.Add(FString(TEXT(""))) + FPackageName::GetAssetPackageExtension());
        Tokens.Add(FString(TEXT(""))) + FPackageName::GetMapPackageExtension());
        Tokens.Add(FString(TEXT(""))) + FPackageName::GetTextAssetPackageExtension());
        Tokens.Add(FString(TEXT(""))) + FPackageName::GetTextMapPackageExtension());

        uint8 PackageFilter = NORMALIZE_DefaultFlags | NORMALIZE_ExcludeEnginePackag

```

If you're **not using version 4.25** then you're good to go, just compile! Otherwise if you're using version 4.25 then one more change is required to fix this engine issue...

Find the `/Engine/Source/Editor/UnrealEd/Private/FileHelpers.cpp` file and search for the `FileHelperPackageUtil::DoesPackageExist(...)` method around line 225:





<pre> bool DoesPackageExist(UPackage* Package, FString* OutFi { // Test using asset registry to figure out existenc IAssetRegistry& AssetRegistry = FAssetRegistryModul if (!AssetRegistry.IsLoadingAssets() !GIsEditor) { TArray<FAssetData> Data; FAssetRegistryModule::GetRegistry().GetAssetsBy if (Data.Num() > 0 && OutFilename) { *OutFilename = FPackageName::LongPackageNam } return Data.Num() > 0; } return FPackageName::DoesPackageExist(Package->GetN } </pre>	<pre> bool DoesPackageExist(UPackage* Package, FString* OutFi { // Test using asset registry to figure out existenc IAssetRegistry& AssetRegistry = FAssetRegistryModul if (!AssetRegistry.IsLoadingAssets() !GIsEditor) { TArray<FAssetData> Data; FAssetRegistryModule::GetRegistry().GetAssetsBy if (Data.Num() > 0 && OutFilename) { *OutFilename = FPackageName::LongPackageNam return true; } return FPackageName::DoesPackageExist(Package->GetN } </pre>
--	---

Just remove `return Data.Num() > 0;` and add a `return true;` instead. Don't forget to put this new line inside the inner if-statement and not where the original return statement was that you deleted, so pay close attention to the braces in the screenshot! *The previous image is only applicable for version 4.25 since it does not exist in any other engine versions.*

Now you're finally good to go and can compile the project!

SAVING

If you try to save a UTXT asset in-editor using version 4.25 you'll experience a really annoying bug where duplicate assets are being saved in both formats. That means you need to use the "clean assets" option under "*Window > JSON*" to remove the broken UTXT files.

Example > Content > MySample				
Name		Date modified	Type	Size
 MyMap.umap		5/27/2020 4:44 PM	UMAP File	635 KB
 MyStruct.uasset		5/7/2021 12:13 AM	UASSET File	7 KB
 NewJsonAsset.uasset		5/9/2021 12:30 PM	UASSET File	2 KB
 NewJsonAsset.utxt		5/9/2021 12:30 PM	UTXT File	3 KB

This is because not only do the engine developers not care to actually finish any features that they implement, but since they never finish these features it's quite common for them to break experimental functionality with their half-assed updates. So if you're using UTXT assets in version 4.25 you essentially have to edit these files in your own text editor while UE4 is closed, and can never use the type-based "structure instance" editor.

However even though the engine developers completely broke this functionality in 4.25 we decided to fix it for you. That's why a precompiled **UE4Editor-UnrealEd.dll** file is included in your download. All you have to do is replace this file in your engine installation. We provided instructions in the previous "cooking" section of this documentation.

NOTE: If you rename or duplicate an asset in the "content browser" it will automatically default to being saved as a binary UASSET. Unfortunately there is no workaround at this time since it's a limitation of the engine design itself. But all you have to do is right-click on these assets after renaming to switch them back to text-based UTXT assets instead.

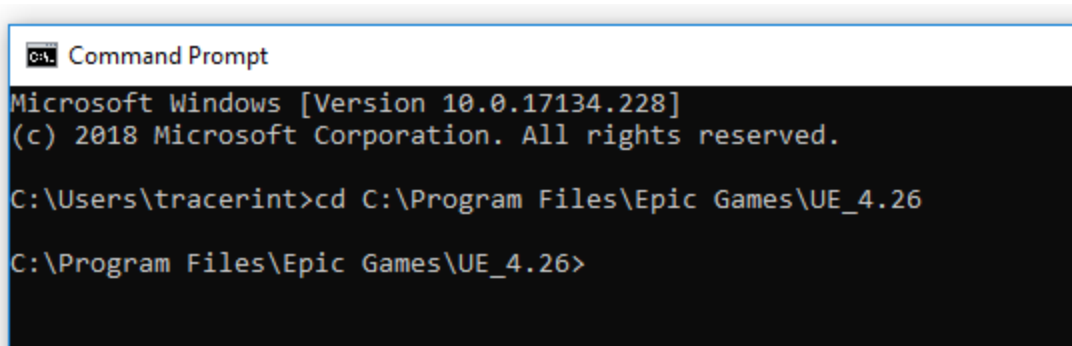
COMPILE

This plugin can be manually compiled for other platforms or engine versions. First open the command prompt (by searching for “cmd” in the start menu) and type the following command:

```
cd "C:\Program Files\Epic Games\UE_4.26"
```

You can copy this command and paste it by right-clicking on the command prompt. Also take note of the **UE_4.26** directory. You need to change this folder to the version that corresponds to the engine version you are using. *If you did not install your engine to the default directory then type the path to your custom installation folder instead.*

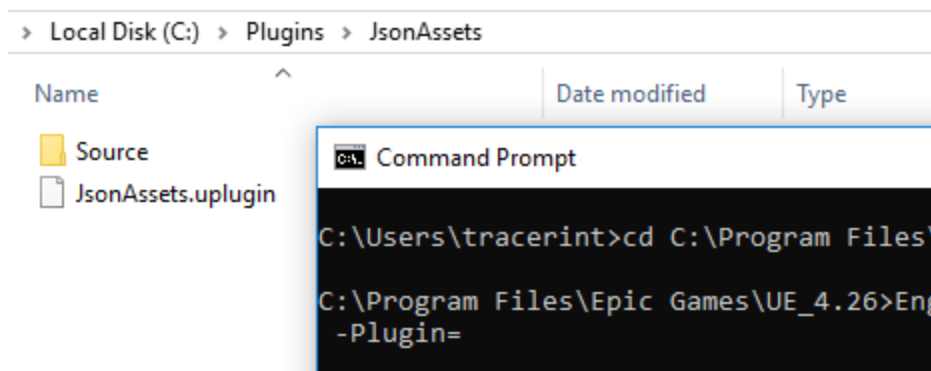
Press ENTER to run the command. You should now see output similar to the following:



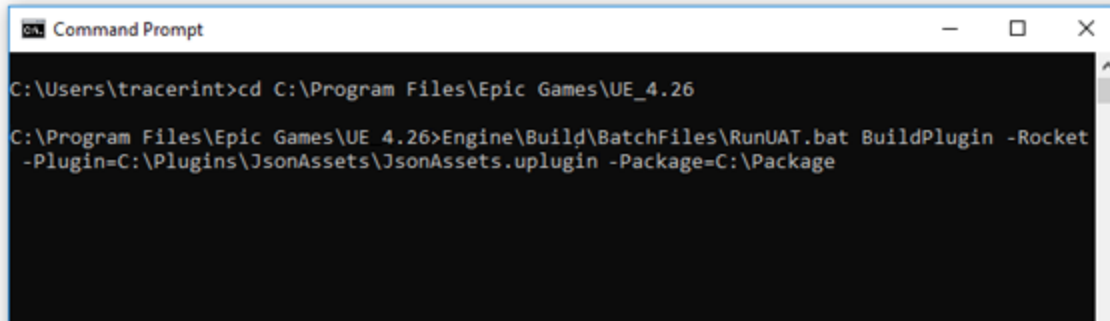
Then type the following command to build the plugin:

```
Engine\Build\BatchFiles\RunUAT.bat BuildPlugin -Rocket -Plugin="..." -Package="..."
```

Replace the first "..." with the path to JsonAssets.uplugin and the last "..." with the path to a temporary “package” folder. You can also drag and drop the .uplugin file or your temporary folder directly onto the command prompt and it will automatically type the path:

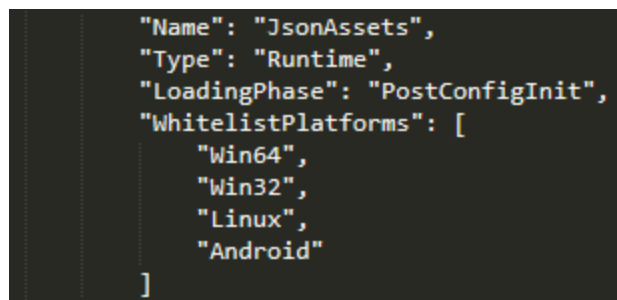


Make sure these paths are **not inside the engine directory** or the build will fail. Once you have the full command typed out it should look similar to the following:



```
Command Prompt
C:\Users\tracerint>cd C:\Program Files\Epic Games\UE_4.26
C:\Program Files\Epic Games\UE_4.26>Engine\Build\BatchFiles\RunUAT.bat BuildPlugin -Rocket
-Plugin=C:\Plugins\JsonAssets\JsonAssets.uplugin -Package=C:\Package
```

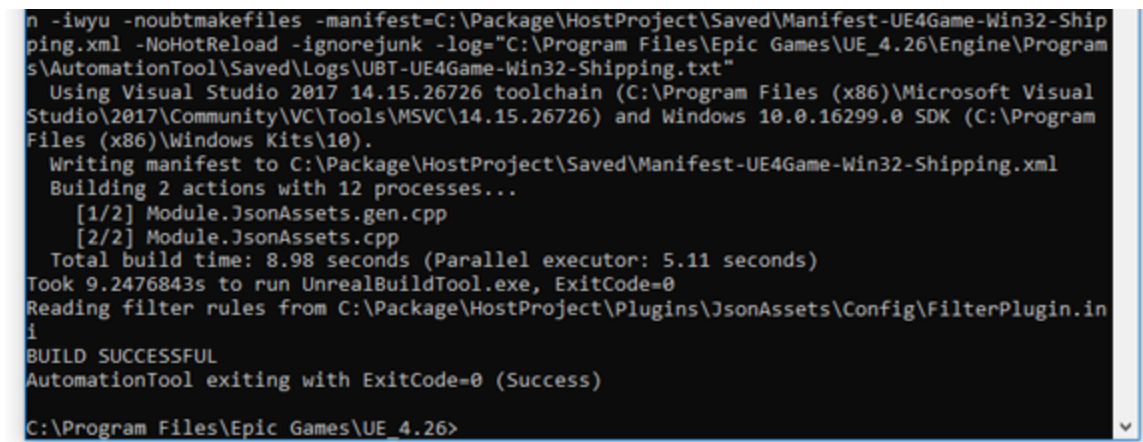
If your machine does not support Mac or Linux builds then you will most likely have to remove the “Mac” and “IOS” or “Linux” platforms from `JsonAssets.uplugin` before compiling:



```
{
  "Name": "JsonAssets",
  "Type": "Runtime",
  "LoadingPhase": "PostConfigInit",
  "WhitelistPlatforms": [
    "Win64",
    "Win32",
    "Linux",
    "Android"
  ]
}
```

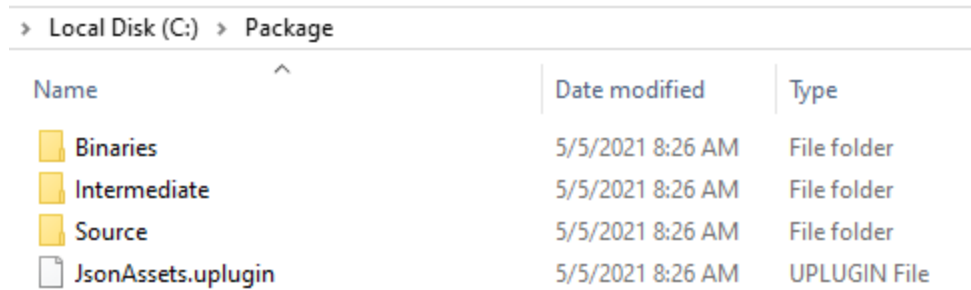
Now hit the ENTER key to run the command. If everything was setup correctly you’ll see many different versions of the plugin being compiled for various platforms.

Once the build is complete you should see the “BUILD SUCCESSFUL” message:



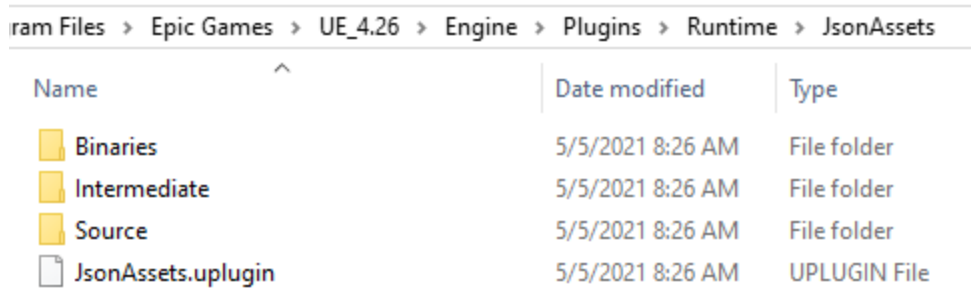
```
n -iwyu -noubtmakefiles -manifest=C:\Package\HostProject\Saved\Manifest-UE4Game-Win32-Shipping.xml -NoHotReload -ignorejunk -log="C:\Program Files\Epic Games\UE_4.26\Engine\Programs\AutomationTool\Saved\Logs\UBT-UE4Game-Win32-Shipping.txt"
Using Visual Studio 2017 14.15.26726 toolchain (C:\Program Files (x86)\Microsoft Visual Studio\2017\Community\VC\Tools\MSVC\14.15.26726) and Windows 10.0.16299.0 SDK (C:\Program Files (x86)\Windows Kits\10).
Writing manifest to C:\Package\HostProject\Saved\Manifest-UE4Game-Win32-Shipping.xml
Building 2 actions with 12 processes...
[1/2] Module.JsonAssets.gen.cpp
[2/2] Module.JsonAssets.cpp
Total build time: 8.98 seconds (Parallel executor: 5.11 seconds)
Took 9.2476843s to run UnrealBuildTool.exe, ExitCode=0
Reading filter rules from C:\Package\HostProject\Plugins\JsonAssets\Config\FilterPlugin.ini
BUILD SUCCESSFUL
AutomationTool exiting with ExitCode=0 (Success)
C:\Program Files\Epic Games\UE_4.26>
```

Check your temporary “package” folder to ensure it looks similar to the following:



Name	Date modified	Type
Binaries	5/5/2021 8:26 AM	File folder
Intermediate	5/5/2021 8:26 AM	File folder
Source	5/5/2021 8:26 AM	File folder
JsonAssets.uplugin	5/5/2021 8:26 AM	UPLUGIN File

Then copy the files in this temporary folder into your engine installation directory. You must do this beforehand if you are compiling any other plugins that require the JsonAssets plugin.



Name	Date modified	Type
Binaries	5/5/2021 8:26 AM	File folder
Intermediate	5/5/2021 8:26 AM	File folder
Source	5/5/2021 8:26 AM	File folder
JsonAssets.uplugin	5/5/2021 8:26 AM	UPLUGIN File

You have now successfully compiled the JsonAssets plugin for your engine version.