Pak Loader

by
Blue Mountains

Contents

Introduction

Pak Loader is a plugin that can be used in Blueprints and C++. Due to how Unreal Engine works .pak files are only intended to be loaded in packaged (shipping) builds.
*While it's technically possible to load .pak files in Editor builds I would highly recommend to **not** do that.*

Mounting a .pak file is easy but accessing the assets within the pak file depends on how the .pak was created and what you are intending to use it for.
Because of this I've created some instructions in this document on how to properly create .pak files, mount and use their assets with this plugin.

You can find an example project and an example .pak file in the link below. Once you have installed this plugin you can open the project and see how it works in action.

UE 4.22 Download
UE 4.23 Download
UE 4.24 Download
UE 4.25 Download
UE 4.26 Download
UE 4.27 Download
UE 5.0 Download

Forum Support Link

In a packaged build you can only mount pak files that contain cooked content. Just using UnrealPak to put a bunch of .uasset files in a pak file won't work unless the content was cooked by the Editor.

Blueprint Functions

C++ functions that are exposed to Blueprints.

In C++ you can use `FPakLoader::Get()` to directly access the pak loader functions.
You can also call the `UPakLoaderLibrary` Blueprint utility static functions.

```
Returns true if this current build is a packaged shipping build.
Returns false if it is an editor build.
bool IsPackagedBuild();


Logs to a file and console, helpful for testing packaged builds to
see what's going on.
@bEnable: true to enable logging.
@NewLogPath: File to output the text log.
void EnableRuntimeLog(bool bEnable, const FString &NewLogPath);


Logs to a file and console, helpful for testing packaged builds to see whats going on.
Requires EnableRuntimeLog to be called first.
@Text: Text to log.
void RuntimeLog(const FString &Text);


Returns name of this Unreal project.
FString GetProjectName();


Returns directory of Unreal's persistent download directory.
FString ProjectPersistentDownloadDir();


Returns SHA1 checksum of a file.
@Filename: File to generate checksum for.
FString SHA1SUM(const FString &Filename);


Filename to packagename. Returns a path starting with a valid root like /Game/, /MyDLC/ etc.
Requires that the path is registered within Unreal. (RegisterMountPoint)
Example: ../../TestProject/Content/Meshes/SM_MyMesh = /Game/Meshes/SM_MyMesh
@Filename: Absolute file or path.
@PackageName: Will hold the converted path. Empty if Filename's path is not registered.
bool TryConvertFilenameToLongPackageName(const FString &Filename, FString &PackageName);


Returns everything after the last slash.
Example: /Game/Textures/T_MyTexture = T_MyTexture
@LongName: Long package name to short package name.
FString GetShortName(const FString &LongName);


Returns an array of all currently mounted pak files.
TArray<FString> GetMountedPakFilenames();
```

```
Tests if a file on disk is a valid .pak file. Also returns size in bytes of the Pak content.
@PakFilename: .pak file on disk.
@PakSize: If pak file is valid then this variable will hold the pak's size in bytes.
bool IsValidPakFile(const FString &PakFilename, int64 &PakSize);


Mounts a pak file and automatically tries to register the mount path.
If you use this function then you don't have to call RegisterMountPoint yourself.
This function also automatically tries to load the asset registry file of the pak.
This only requirement for this function is that the pak file contains an AssetRegistry.bin file
in order to detect the root and content path automatically.
@PakFilename: .pak file on disk.
bool MountPakFileEasy(const FString &PakFilename);


Mounts a pak file. !!!Read the plugins documentation to learn about mount points etc.!!!
@PakFilename: .pak file on disk.
@MountPath: Where to mount the Pak content. Leave empty if unsure (mount path as specified in
the pak file will be used).
bool MountPakFile(const FString &PakFilename, const FString &MountPath);


Unmounts a Pak that was previously mounted.
@PakFilename: .pak file on disk to unmount.
bool UnmountPakFile(const FString &PakFilename);


Creates a link between a root path and a package content path (mount point).
This is required to make references between assets work. Should be called after mounting a pak.
See the parameters below for an example. After calling this function, the RootPath will point
to the ContentPath.
Without registering a mount point the engine doesn't know the root path of your mounted plugin.
The ContentPath is the mount point of the pak file + the further part to your "Content" folder.
See this image for another example:
https://drive.google.com/file/d/1jKlOshFSnXQIXwKyENl1zwsskhl4Vke2/view?usp=sharing
@RootPath: Top content folder name of where your assets are in (Example: /TestDLC/).
@ContentPath: The path inside a pak to where the RootPath should point to. (Example:
../../../TestProject/Plugins/TestDLC/Content/)
void RegisterMountPoint(const FString &RootPath, const FString &ContentPath);


Unregister previously registered mount point. See RegisterMountPoint function.
void UnRegisterMountPoint(const FString &RootPath, const FString &ContentPath);


Loads an AssetRegistry.bin file to publish new files to Unreal's asset registry.
@AssetRegistryFile: Full path to an AssetRegistry.bin file. Example:
(../../../TestProject/Plugins/TestDLC/AssetRegistry.bin)
void LoadPakAssetRegistryFile(const FString &AssetRegistryFile);


Registers an AES encryption key to the engine.
@Guid: The encryption key guid. For example 00000000000000000000000000000000
@AesKey: The AES key as base64. For example: zLQDKoikfG77t0B84QGt8CTpAyVjjj86dX3vo8mEmUE=
bool RegisterEncryptionKey(const FString &Guid, const FString &AesKey);


Returns all files in a specific pak directory.
@PakDirectory: Path in pak (Example: ../../../TestProject)
@bRecursively: true to also return files in subfolders.
TArray<FString> GetFilesInPakDirectory(const FString &PakDirectory, bool bRecursively);


Returns all files from inside a .pak file
```

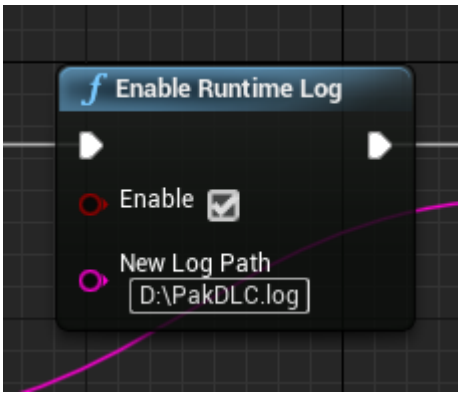
```
@PakFilename: .pak file on disk.
TArray<FString> GetFilesInPak(const FString &PakFilename, bool bUAssetOnly = true);


Tests if a specific pak directory exists.
bool DoesPakDirectoryExist(const FString &PakDirectory);



Loads any class (ie Blueprints) from a pak file. Returns UClass which you can cast to your
desired class (ie AActor).
@Filename: The file to load as class. (Example: /TestDLC/Blueprints/BP_Test)
UClass *GetPakFileClass(const FString &Filename);


Loads any object (assets) from a pak file. Returns UObject which you can cast to your desired
asset class type.
@Filename: The file to load as object. (Example: /TestDLC/Meshes/SM_Chair)
UObject *GetPakFileObject(const FString &Filename);


Utility to load UTexture2D asset from pak.
UTexture2D *GetPakFileTexture2D(const FString &Filename);


Utility to load UStaticMesh asset from pak.
UStaticMesh *GetPakFileStaticMesh(const FString &Filename);


Utility to load USkeletalMesh asset from pak.
USkeletalMesh *GetPakFileSkeletalMesh(const FString &Filename);


Utility to load USoundBase asset from pak.
USoundBase *GetPakFileSound(const FString &Filename);


Utility to load UMaterial asset from pak.
UMaterial *GetPakFileMaterial(const FString &Filename);


Utility to load UMaterialInstanceConstant asset from pak.
UMaterialInstanceConstant *GetPakFileMaterialInstanceConstant(const FString &Filename);


Utility to load UAnimSequence asset from pak.
UAnimSequence *GetPakFileAnimSequence(const FString &Filename);


Reads content as string from pak. Requires full absolute path.
bool GetPakFileText(const FString &Filename, FString &String);
```

Using the example project

You will need to have the Pak Loader plugin installed for your engine version.

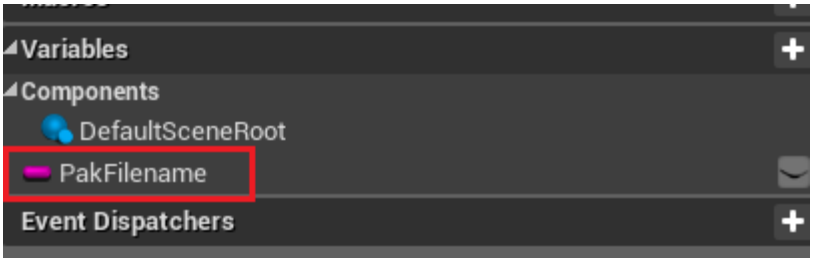
Step 1

Open the Blueprint “BP_PakLoading”
Modify the log path to a valid path.



Step 2

Modify the *PakFilename* variable to point to the example .pak file.



Step 3

Package the project:
File -> Package Project -> Windows -> Windows (64-bit)

Once finished go to the packing directory and launch the newly created PakDemoProject.exe.
You will see a chair and hear a fire sound that have been loaded from the example pak file.
Open the log file to see what’s going on.

Check the “BP_PakLoading” Blueprint which shows example usage of the provided functions.

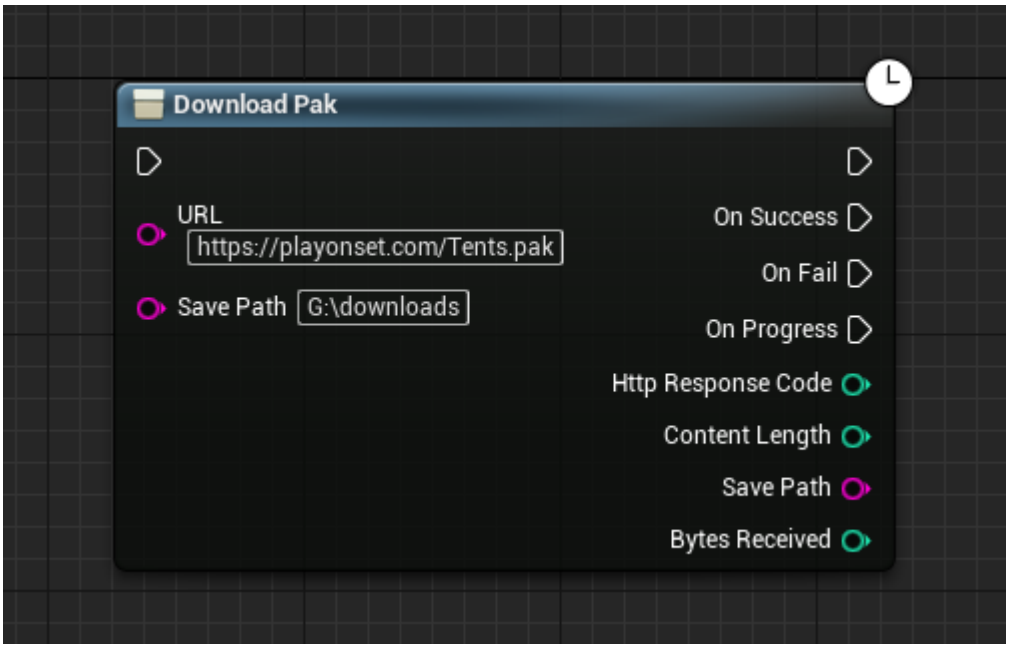
Downloading .pak files via HTTP

PakLoader provides a download function that can download any file from HTTP servers.

URL: Link to your pak file.
Save Path: Location to save the pak file to.

This function is asynchronous meaning there are two callbacks that are being fired once the download completes or an error occurs.

HttpResponseCode: The HTTP response code returned from the server. 200 on success.
ContentLength: Number of bytes downloaded and saved. (0 on fail)
Save Path: The same value that you passed when calling this function. (Empty on fail)
BytesReceived: Number of bytes received so far in OnProgress callback.



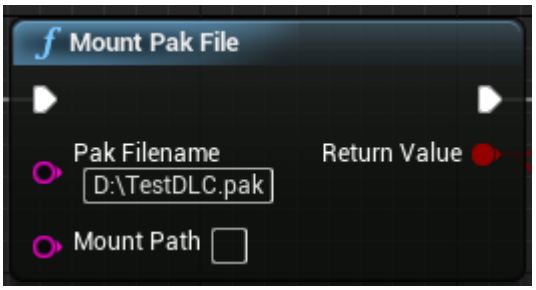
Mounting pak files and loading assets

At first, we want to view the contents of a pak file to get aware of its structure. Open a command prompt and go to your Engine's directory.

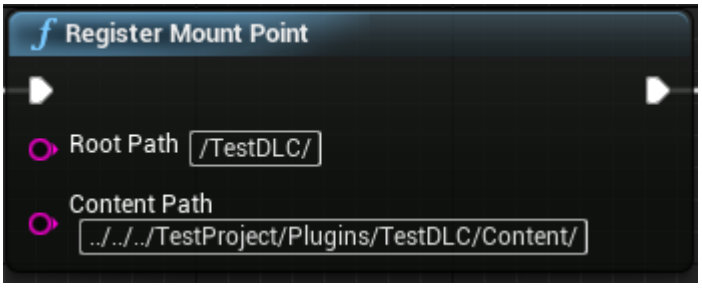
```
cd "C:\Program Files\Epic Games\UE_4.22\Engine\Binaries\Win64"
UnrealPak.exe D:\Path\MyPakFile.pak -List
```

```
C:\Program Files\Epic Games\UE_4.22\Engine\Binaries\Win64>UnrealPak.exe D:\TestDLC.pak -List
LogPakFile: Display: Using command line for crypto configuration
LogPakFile: Display: Added 0 entries to add to pak file.
LogPakFile: Display: Mount point ../../../
LogPakFile: Display: "Engine/Content/EditorResources/EmptyActor.uasset" offset: 0, size: 531 byt
LogPakFile: Display: "Engine/Content/EditorResources/LightIcons/S_LightError.uasset" offset: 601
LogPakFile: Display: "Engine/Content/EditorResources/EmptyActor.uexp" offset: 2048, size: 25865
LogPakFile: Display: "Engine/Content/EditorResources/LightIcons/S_LightError.uexp" offset: 2867
LogPakFile: Display: "TestProject/Plugins/TestDLC/AssetRegistry.bin" offset: 40960, size: 3536 b
LogPakFile: Display: "TestProject/Plugins/TestDLC/Config/FilterPlugin.ini" offset: 44566, size:
LogPakFile: Display: "TestProject/Plugins/TestDLC/Content/Audio/Fire01.uasset" offset: 45056, si
LogPakFile: Display: "TestProject/Plugins/TestDLC/Content/Audio/Fire01.uexp" offset: 45719, size
LogPakFile: Display: "TestProject/Plugins/TestDLC/Content/Audio/Fire01_Cue.uasset" offset: 1502
LogPakFile: Display: "TestProject/Plugins/TestDLC/Content/Audio/Fire01_Cue.uexp" offset: 150843,
LogPakFile: Display: "TestProject/Plugins/TestDLC/Content/Blueprints/BP_Test.uasset" offset: 151
LogPakFile: Display: "TestProject/Plugins/TestDLC/Content/Blueprints/BP_Test.uexp" offset: 1540
LogPakFile: Display: "TestProject/Plugins/TestDLC/Content/Maps/DLCMap.uexp" offset: 154792, size
LogPakFile: Display: "TestProject/Plugins/TestDLC/Content/Maps/DLCMap.umap" offset: 155648, size
LogPakFile: Display: "TestProject/Plugins/TestDLC/Content/Materials/M_Chair.uasset" offset: 157
LogPakFile: Display: "TestProject/Plugins/TestDLC/Content/Materials/M_Chair.uexp" offset: 158984
LogPakFile: Display: "TestProject/Plugins/TestDLC/Content/Materials/M_Color.uasset" offset: 178
LogPakFile: Display: "TestProject/Plugins/TestDLC/Content/Materials/M_Color.uexp" offset: 17886
LogPakFile: Display: "TestProject/Plugins/TestDLC/Content/Materials/MI_Table.uasset" offset: 19
LogPakFile: Display: "TestProject/Plugins/TestDLC/Content/Materials/MI_Table.uexp" offset: 1976
LogPakFile: Display: "TestProject/Plugins/TestDLC/Content/Meshes/SM_Chair.uasset" offset: 19865
LogPakFile: Display: "TestProject/Plugins/TestDLC/Content/Meshes/SM_Chair.uexp" offset: 199950,
LogPakFile: Display: "TestProject/Plugins/TestDLC/Content/Meshes/SM_Table.uasset" offset: 29696
LogPakFile: Display: "TestProject/Plugins/TestDLC/Content/Meshes/SM_Table.uexp" offset: 298286,
LogPakFile: Display: "TestProject/Plugins/TestDLC/Content/Textures/T_Chair_M.uasset" offset: 54
LogPakFile: Display: "TestProject/Plugins/TestDLC/Content/Textures/T_Chair_M.ubulk" offset: 5468
LogPakFile: Display: "TestProject/Plugins/TestDLC/Content/Textures/T_Chair_M.uexp" offset: 97484
LogPakFile: Display: "TestProject/Plugins/TestDLC/Content/Textures/T_Chair_N.uasset" offset: 978
LogPakFile: Display: "TestProject/Plugins/TestDLC/Content/Textures/T_Chair_N.ubulk" offset: 9809
LogPakFile: Display: "TestProject/Plugins/TestDLC/Content/Textures/T_Chair_N.uexp" offset: 3297
LogPakFile: Display: "TestProject/Plugins/TestDLC/Content/Textures/T_TableRound_M.uasset" offset
LogPakFile: Display: "TestProject/Plugins/TestDLC/Content/Textures/T_TableRound_M.ubulk" offset:
LogPakFile: Display: "TestProject/Plugins/TestDLC/Content/Textures/T_TableRound_M.uexp" offset:
LogPakFile: Display: "TestProject/Plugins/TestDLC/Content/Textures/T_TableRound_N.uasset" offset
LogPakFile: Display: "TestProject/Plugins/TestDLC/Content/Textures/T_TableRound_N.ubulk" offset:
LogPakFile: Display: "TestProject/Plugins/TestDLC/Content/Textures/T_TableRound_N.uexp" offset:
LogPakFile: Display: "TestProject/Plugins/TestDLC/Metadata/DevelopmentAssetRegistry.bin" offset:
LogPakFile: Display: 37 files (5646901 bytes), (0 filtered bytes).
LogPakFile: Display: Unreal pak executed in 0.005426 seconds
```

Leave the Mount Path parameter empty. The mount point from the pak file will be used (../../../).



Create a RootPath that points to our ContentPath in the pak file.



Now the engine knows that the root path `/TestDLC/` points to the content path `../../../TestProject/Plugins/TestDLC/Content/`

Meaning we can now access assets like this:
/TestDLC/Audio/Fire01_Cue
/TestDLC/Blueprints/BP_Test
/TestDLC/Meshes/SM_Chair
etc…

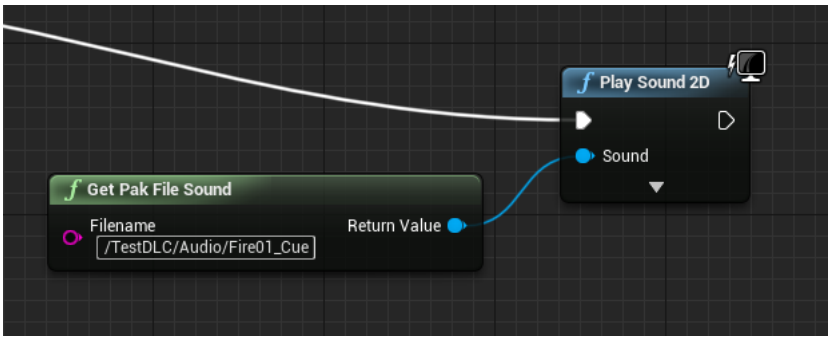
Since we registered the mount point above, all references are intact. For example, a static mesh that references materials which references textures will now work.

Example how the RootPath and ContentPath assembles:

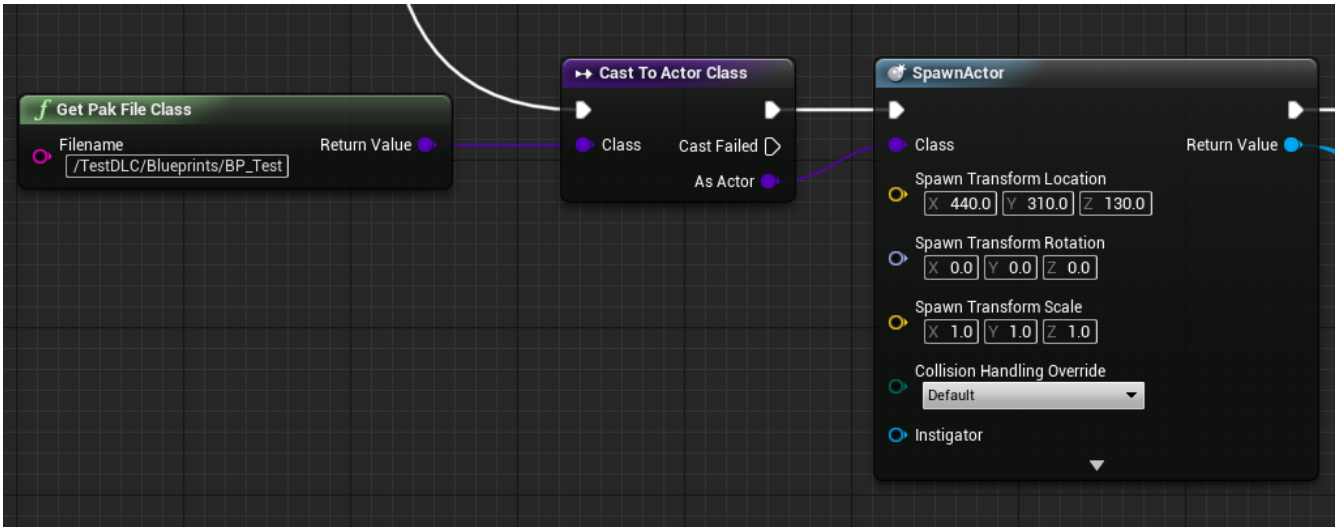
```
C:\Program Files\Epic Games\UE_4.27\Engine\Binaries\Win64>UnrealPak -List "                                    \PakDemoProject\TestDLC.pak"
LogInit: Display: Loading text-based GConfig....
LogPakFile: Display: Using command line for crypto configuration
LogPakFile: Display: Mount point ../../../ 2
LogPakFile: Display: "Engine/Content/Functions/Engine_MaterialFunctions02/Utility/BlendAngleCorrectedNormals.uasset" offset: 0, size: 811 bytes, sha1: D2FD74FBEDFDD31F9107EDF5542A67B
LogPakFile: Display: "Manifest_NonUFSFiles_Win64.txt" offset: 864, size: 70 bytes, sha1: DC865B1176F10A57B335F770BBFC4FABCDC18716, compression: None.
LogPakFile: Display: "Manifest_UFSFiles_Win64.txt" offset: 987, size: 412 bytes, sha1: FE72A4ACBC5FCBCA182AA1281683DF20B5507011, compression: Zlib.
LogPakFile: Display: "TestProject/Plugins/TestDLC/AssetRegistry.bin" offset: 1472, size: 2852 bytes, sha1: 8417DF9994C097B1A6526469A8E989EC36FD5912, compression: Zlib.
LogPakFile: Display: "TestProject/Plugins/TestDLC/Content/Audio/Fire01_Cue.uasset" offset: 4397, size: 801 bytes, sha1: 7AC03406667527FD95447422DBFAFDC298169113, compression: None.
LogPakFile: Display: "TestProject/Plugins/TestDLC/Content/Blueprints/BP_Test.uasset" offset: 5251, size: 930 bytes, sha1: 307BDED2F9AFCD90EB2DC6C0F0089844F6C35605, compression: Zlib.
LogPakFile: Display: "TestProject/Plugins/TestDLC/Content/Materials/M_Color.uasset" offset: 6254, size: 1262 bytes, sha1: 5E08E4D4F39029D9682C49A02EBAC26BCDCBA671, compression: None.
LogPakFile: Display: "TestProject/Plugins/TestDLC/Content/Materials/M_Statue.uasset" offset: 7569, size: 1310 bytes, sha1: EE54741F426A18C9283474AA51CCE0AAF27CC532, compression: Zlib
LogPakFile: Display: "TestProject/Plugins/TestDLC/Content/Materials/M_Statue_Inst.uasset" offset: 8952, size: 1120 bytes, sha1: A9E594EB2E01D693BDD881F66BDF32679AA52734, compression:
LogPakFile: Display: "TestProject/Plugins/TestDLC/Content/Meshes/SM_Statue.uasset" offset: 10145, size: 1291 bytes, sha1: E90A3566EB66E87504558CF8487024AD12F995DE, compression: Zlib.
LogPakFile: Display: "TestProject/Plugins/TestDLC/Content/Textures/T_Crackle.uasset" offset: 11509, size: 737 bytes, sha1: 918A62E2404A2BB91DFC6F4C583304052BD8E1BA, compression: None
```

```
RegisterMountPoint(RootPath, ContentPath)

        RootPath = 1 (/TestDLC/)
        ContentPath = 2 + 3 (../../../TestProject/Plugins/TestDLC/Content/)
```

Example play sound from pak file:



Spawn Blueprint from pak file:

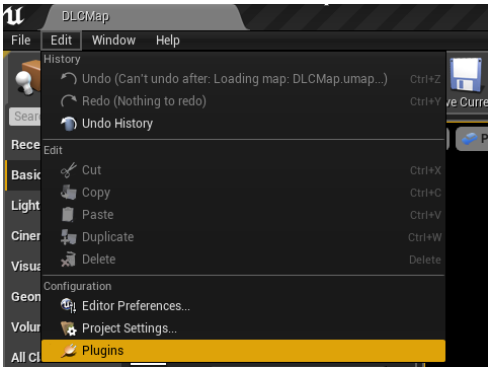


Creating a DLC .pak file. (Standalone .pak file with only necessary content and working references)

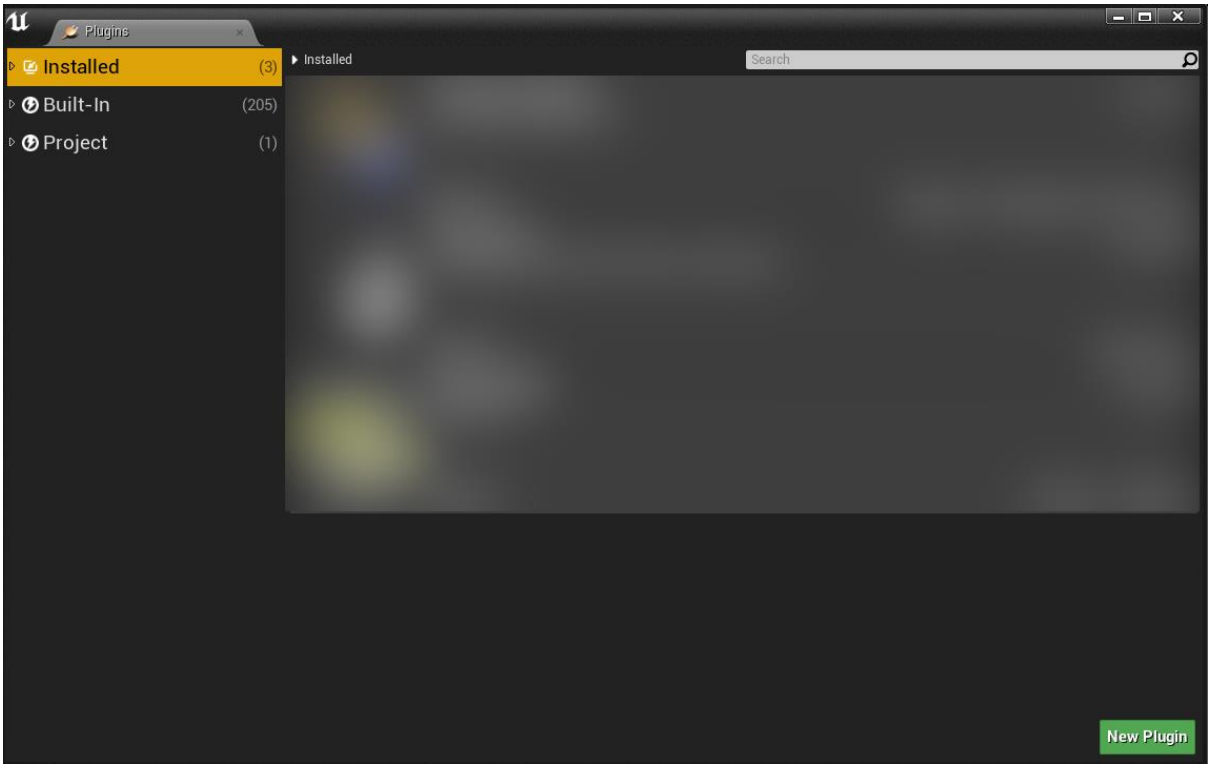
Create a new Unreal Engine project and name it “TestProject” for the sake of this tutorial.

Before you continue you should disable the “Share Material Shader Code” setting in your project settings. Also disable it in your other project where you load the pak file from. In case you would like to use the shared material library you can try to load the pak with the “MountPakFileEasy” function.

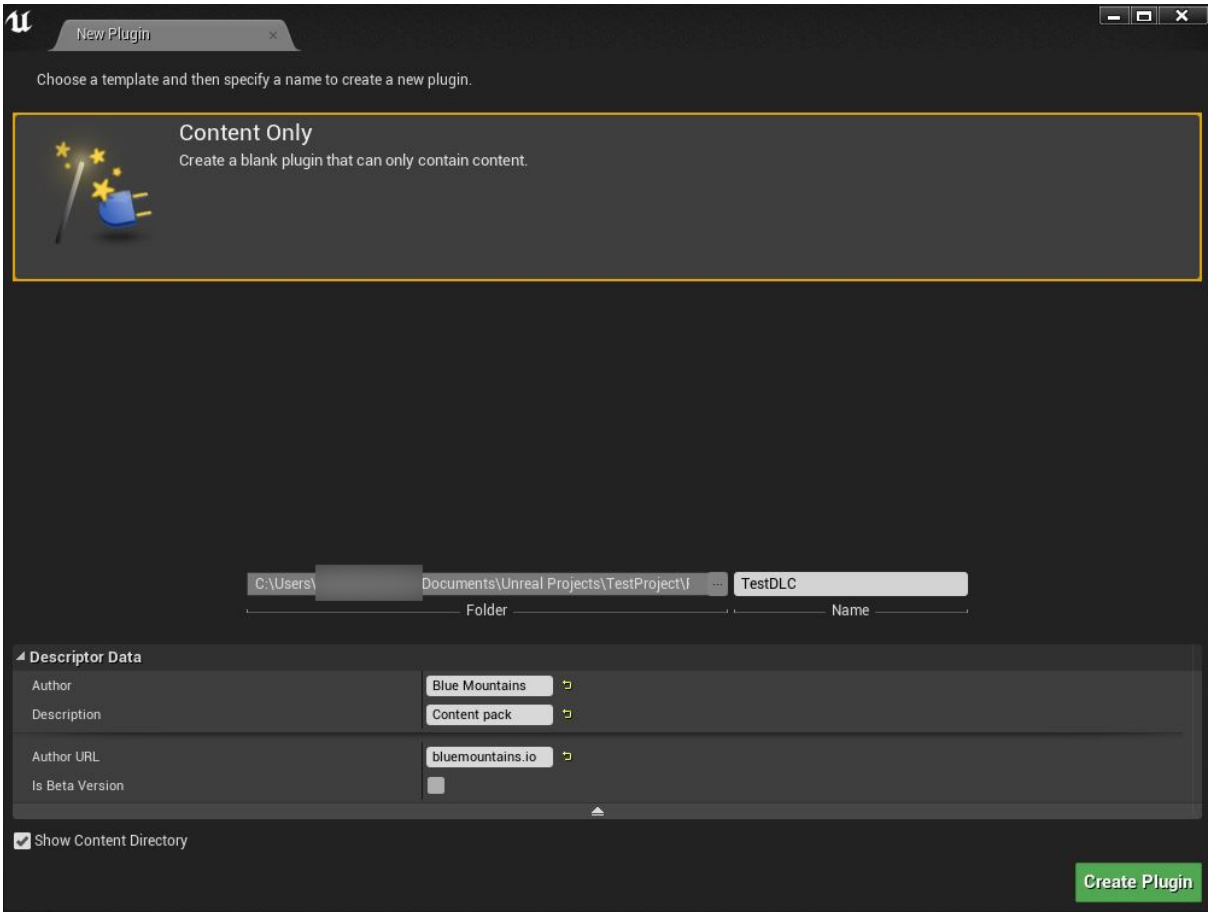
- Edit-> Plugins



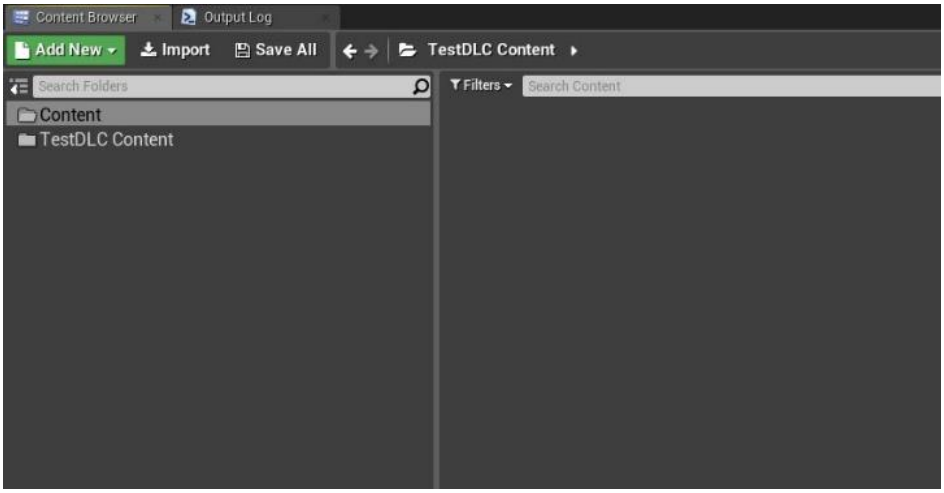
- New Plugin



- Select “Content Only”
- Enter a name for the DLC. In this example “TestDLC”.
- Enter author, description, etc.
- Finally click Create Plugin.

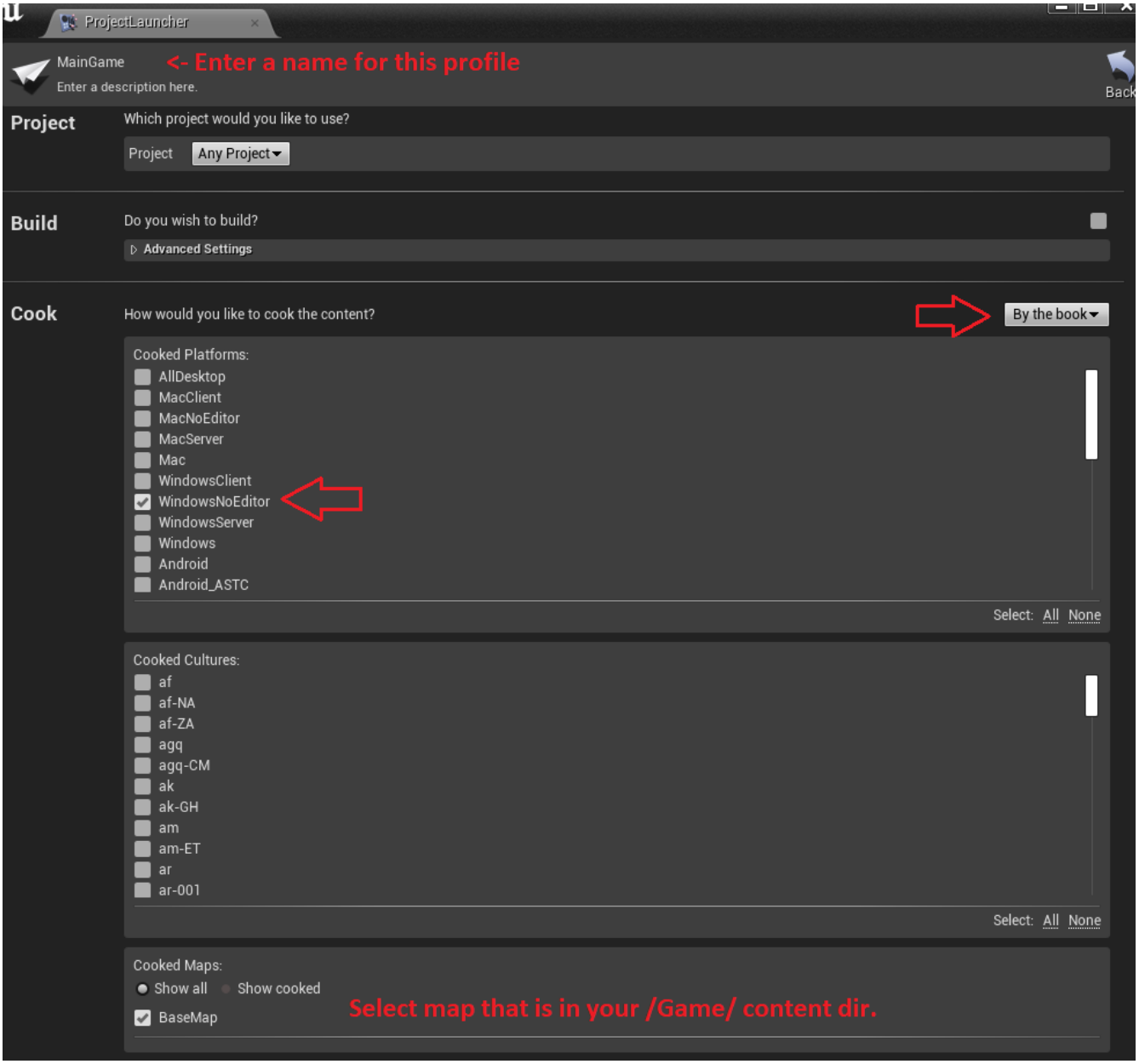


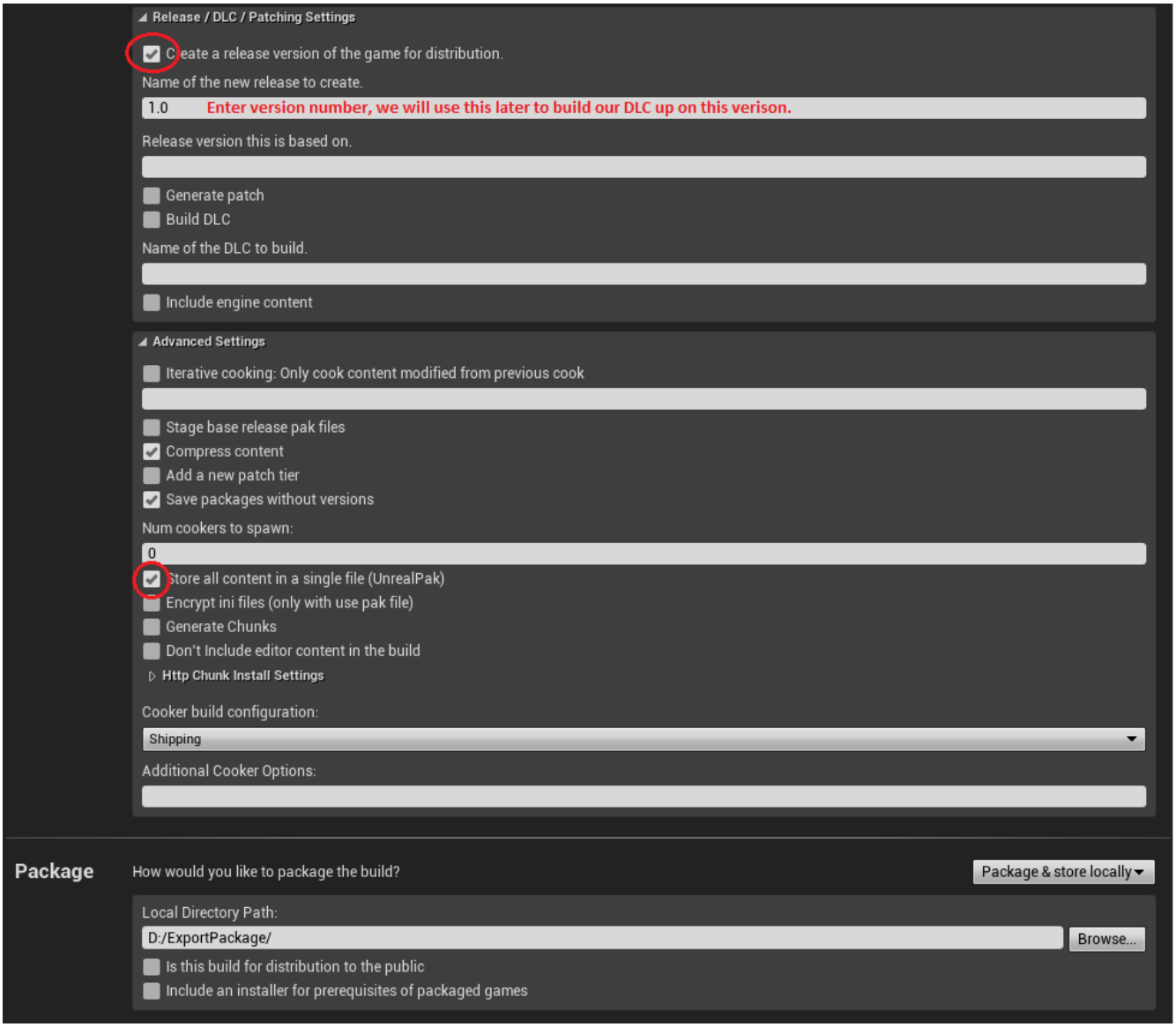
You will have a new Content folder in the Content Browser named “TestDLC”. You can place all your files in this directory.
This will be your new root path for all DLC assets. Instead of /Game/ for the normal game content root the new root path is /TestDLC/[ExampleAsset].
This is very important as we can later mount this DLC pak with /TestDLC/ without breaking references and without messing content in /Game/.



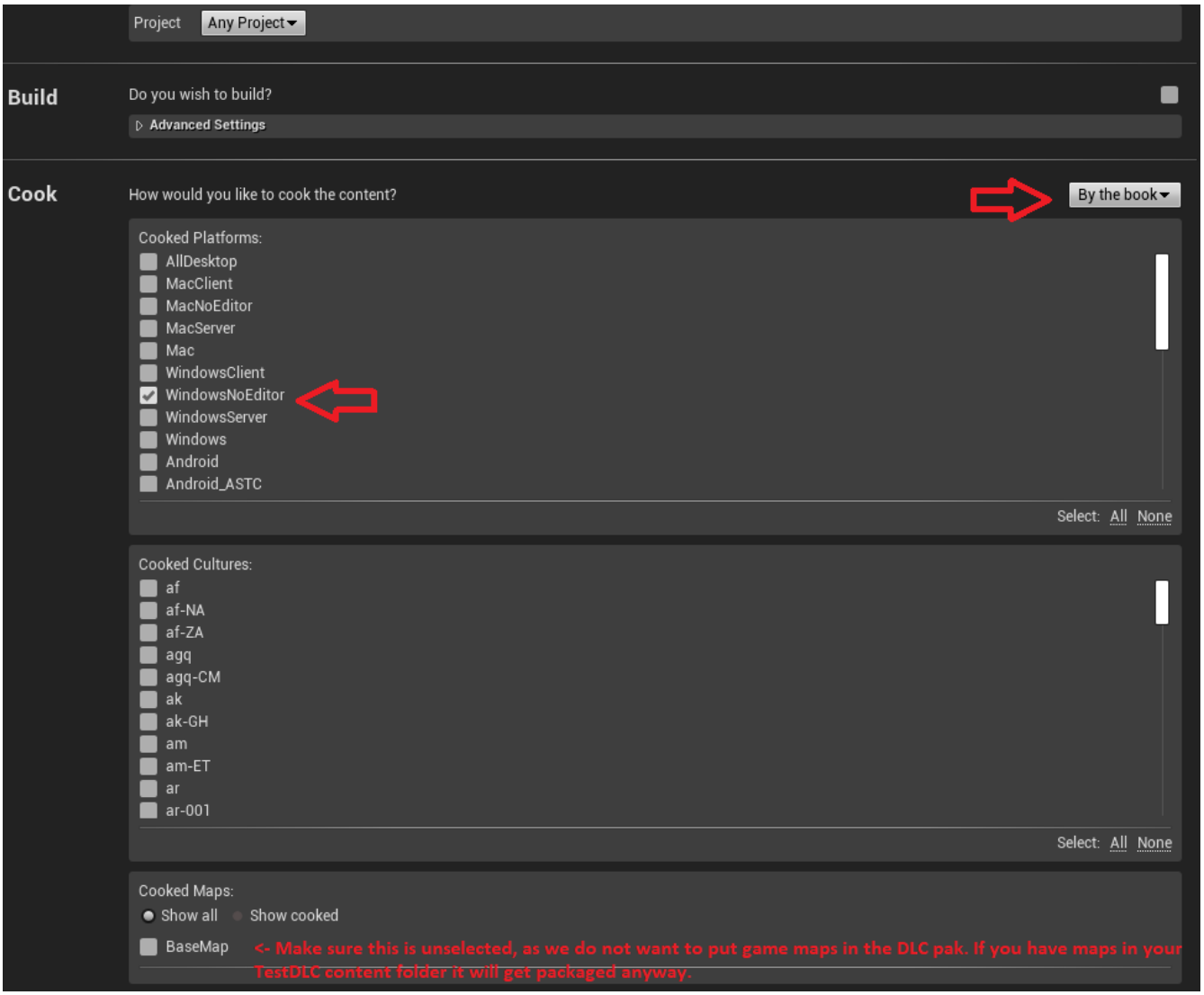
Window -> Project Launcher

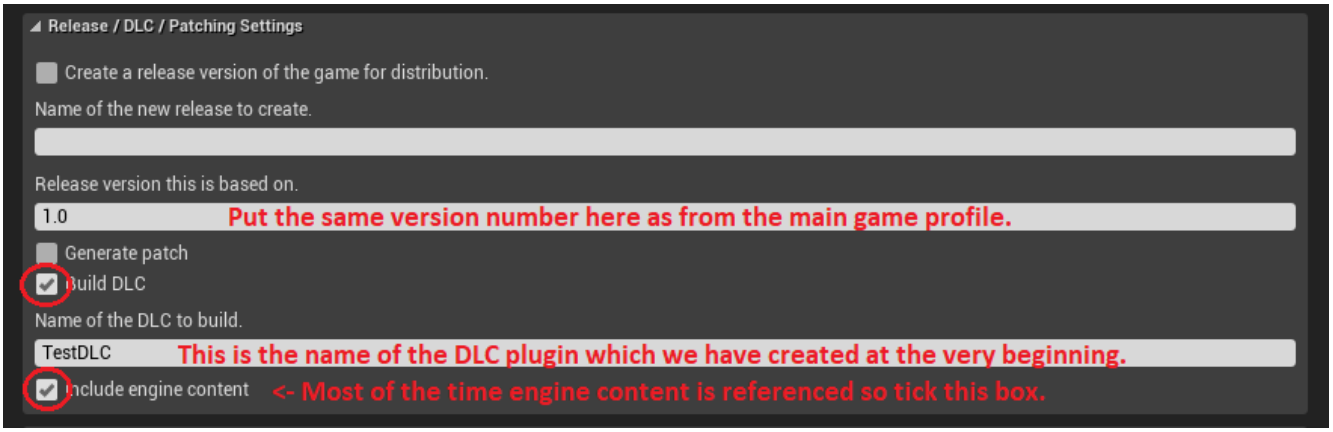
We need to create two profiles. One for packaging the game (“MainGame”) and second one for our DLC pak file (“TestDLC”).

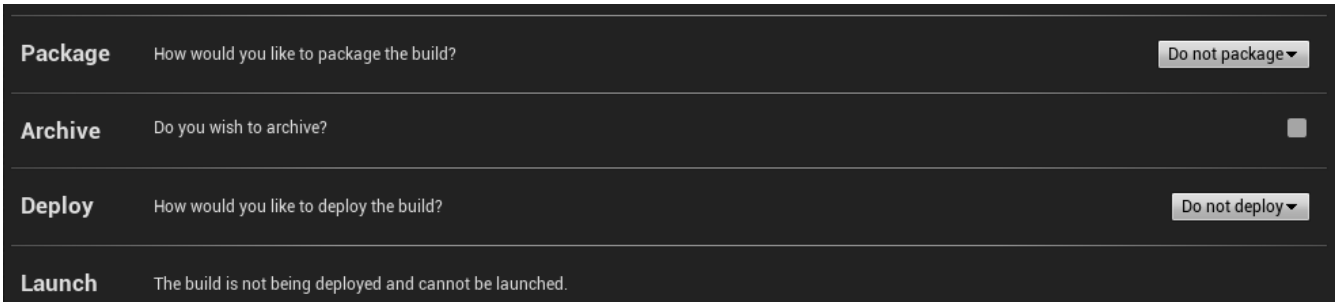




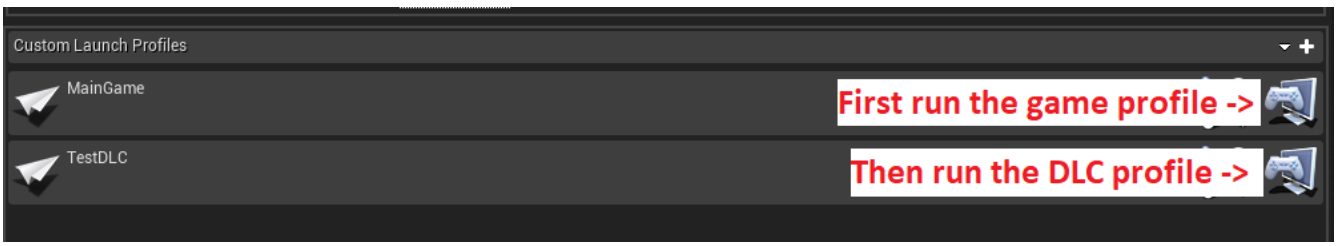
Now go back and edit the second profile “TestDLC”.


Project
Any Project
Build
Do you wish to build?
Advanced Settings
Cook
How would you like to cook the content?
By the book
Cooked Platforms:
AllDesktop
MacClient
MacNoEditor
MacServer
Mac
WindowsClient
WindowsNoEditor
WindowsServer
Windows
Android
Android_ASTC
Select: All None
Cooked Cultures:
af
af-NA
af-ZA
agq
agq-CM
ak
ak-GH
am
am-ET
ar
ar-001
Select: All None
Cooked Maps:
Show all
Show cooked
BaseMap
<- Make sure this is unselected, as we do not want to put game maps in the DLC pak. If you have maps in your TestDLC content folder it will get packaged anyway.


Release / DLC / Patching Settings
Create a release version of the game for distribution.
Name of the new release to create.
Release version this is based on.
1.0
Put the same version number here as from the main game profile.
Generate patch
Build DLC
Name of the DLC to build.
TestDLC
This is the name of the DLC plugin which we have created at the very beginning.
Include engine content
<- Most of the time engine content is referenced so tick this box.



All profiles are now correctly setup.
Now run the game profile to package the project in version 1.0.
Then run the DLC profile which will create a pak file that only contains content /TestDLC.



The DLC .pak file will be this directory:

```
C:\Users\X\Documents\UnrealProjects\TestProject\Plugins\TestDLC\Saved\StagedBuilds\WindowsNoEditor\TestProject\Plugins\TestDLC\Content\Paks\WindowsNoEditor\TestDLCTestProject-WindowsNoEditor.pak
```

UE5 Notes

When using Unreal Engine 5 make sure that “Use Io Store” setting is disabled in your project settings.

Changelog

1.0 – Aug 31 2019
Initial release.

1.1 – Jul 06 2020
Add GetFilesInPak function.
Improve IsPackagedBuild.
Add RegisterEncryptionKey function.

1.2 – Nov 29 2020
Add OnProgress callback for DownloadPak.
If the user did not specify a filename, try to extract it from the download URL.
Support Linux and Mac in uplugin file.

Dec 8, 2020
Support for 4.26

Aug 4, 2021
Support for 4.27 and 5.0EA (4.27 code is compatible to 5.0EA, MP does not support downloading for 5.0 yet)

1.3 - Dec 26 2021 (UE 4.25-4.27)
IsPackagedBuild now uses checks at runtime rather than compile time.
Whitelist for Mac, Android and IOS

1.4 – Feb 23 2021 (UE 4.27)
Add function “MountPakFileEasy”

Troubleshooting

Problem: An asset that I load from the pak file does not show up/work in packaged builds. But it works in Editor.
Solution: Use the UnrealPak.exe tool from the engine binaries folder to output the contents of your pak file. If the said asset is not inside the pak then it probably got not packaged by Unreal. Check if the asset correctly resides inside your plugin content directory. Double check of references between your assets.
Also check if you have any “Redirectors”. Content Browser -> Other Filters -> Show Redirectors.

Problem: I get X error message when packaging a pak file in Unreal.
Solution: Delete all temporary files in your project. Double check on Project Launcher settings from this documentation. Then try again. If the problem persists you have to ask Epic Games or file a bug report to Epic Games. You can ask on the marketplace, maybe I know the error message.