GAME FEATURES MODDING GUIDE

Introduction

With the release of Zone Kit Phase 2, mods are no longer just plain .uplugin-based plugins. They are now game features.

Here's an overview of what you can do from now on when creating new mods or after converting older ones to the new format:

- Add new actors to the world that support World Partition and level streaming.
- Properly add new assets and packages, e.g., new custom materials, blueprint classes, etc.:
    - Each mod game feature, when appropriate, will package its own asset registry as well as a shader library, which will be processed by the game at runtime.
    - Each mod game feature will now have a corresponding mounting path while in-game, e.g., Stalker2/Mods/<ModName>, eliminating any asset collisions between mods.
- Etc.

Let's look at what has been changed and what you should be aware of in detail.

Legacy mods

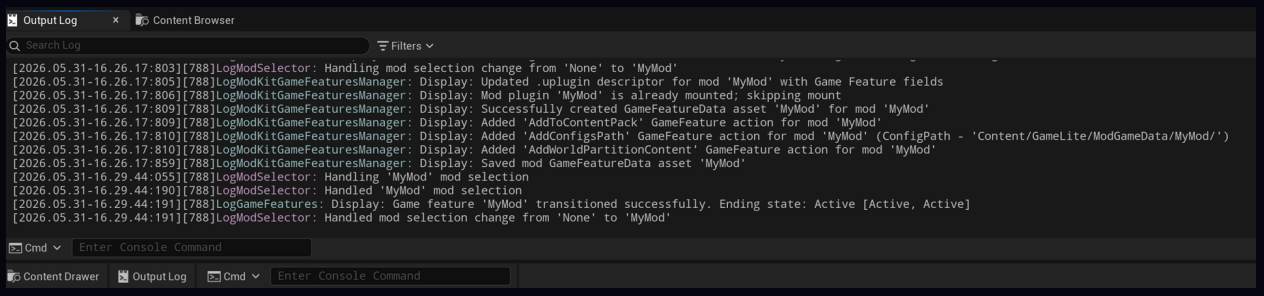
While we have tried to ensure that legacy .uplugin-based mods still function properly in-game, you will have to update your existing mods to continue working with them in Zone Kit. Fortunately, the process of converting an older mod to a new format is seamless and happens automatically when you select a mod in Zone Kit.

Meet ModKitGameFeaturesManager. This is an entity that ensures the following:

1. New mods are created as mod game features.

2. Existing mods are seamlessly updated.

The corresponding LogModKitGameFeaturesManager log category is there to help you track these operations.

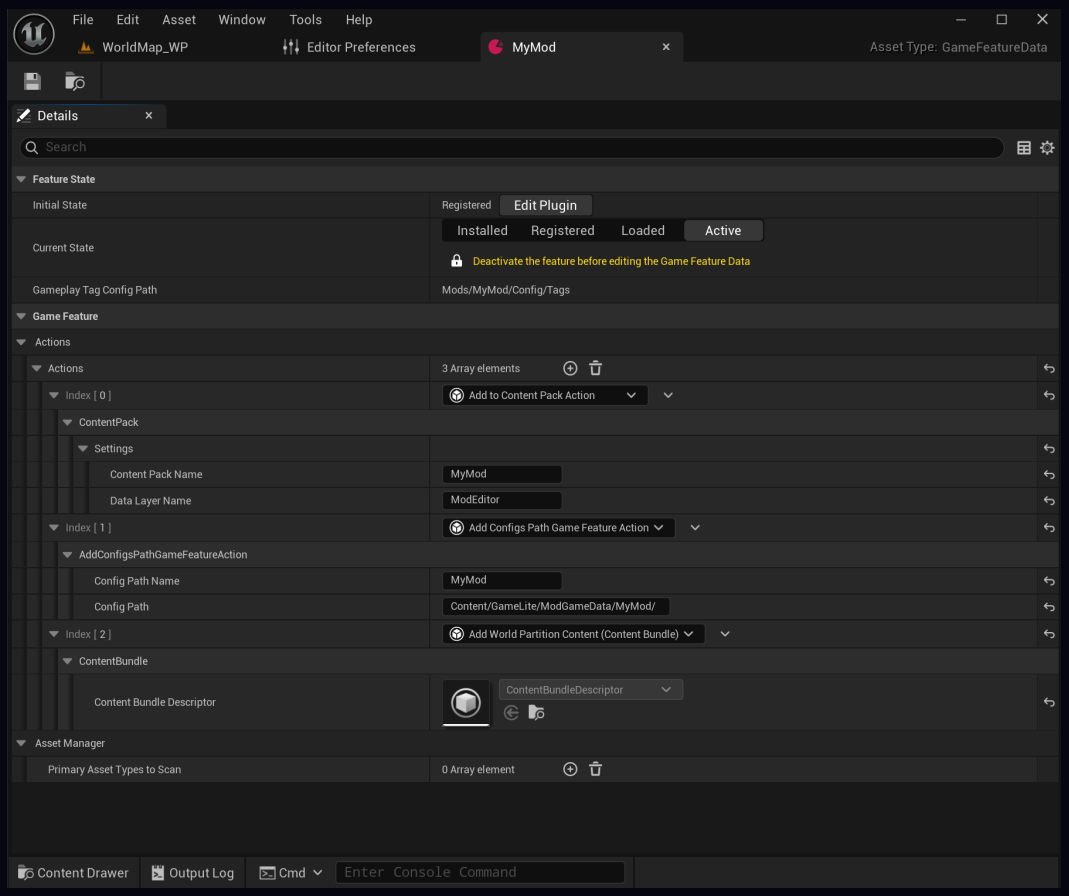


And should any errors occur, corresponding pop-ups will inform you about what exactly went wrong.

Working with mod game features

Nothing has changed when it comes to actually working with mods in Zone Kit – checking out assets, creating new ones, etc. The only difference is the extended capabilities in terms of what you can now do, which have already been mentioned above. However, a couple of things are still worth noting.

Firstly, you will notice that a new asset, which shares the mod's name, is present in the mod's content directory. It's a Game Feature Data Asset, which contains necessary game feature actions for the mod to be registered and function properly while in-game. The asset is created and updated programmatically.



WARNING

DO NOT modify the asset or the predefined game feature actions unless you know what you are doing, as it may lead to the mod not working properly.

Secondly, you may notice that the .uplugin descriptor for the mod is still present and now has additional fields. This is another crucial piece for mods to work properly and be processed as game features.

Here's a comparison table between a legacy and mod game feature .uplugin descriptor files:

Legacy .uplugin-only mod	Mod game feature

<pre>{
"FileVersion": 3,
"Version": 1,
"VersionName": "1.0",
"FriendlyName": "MyMod",
"Description": "",
"Category": "Other",
"CreatedBy": "",
"CreatedByURL": "",
"DocsURL": "",
"MarketplaceURL": "",
"SupportURL": "",
"CanContainContent": true,
"IsBetaVersion": false,
"IsExperimentalVersion": false,
"Installed": false
}</pre>
	<pre>{
	"FileVersion": 3,
	"Version": 1,
	"VersionName": "1.0",
	"FriendlyName": "MyMod",
	"Description": "",
	"Category": "Game Features",
	"CreatedBy": "",
	"CreatedByURL": "",
	"DocsURL": "",
	"MarketplaceURL": "",
	"SupportURL": "",
	"CanContainContent": true,
	"IsBetaVersion": false,
	"IsExperimentalVersion": false,
	"Installed": false,
	"EnabledByDefault": true,
	"ExplicitlyLoaded": true,
	"BuiltInInitialFeatureState":
	"Registered",
	"Mod": true
	}</pre>

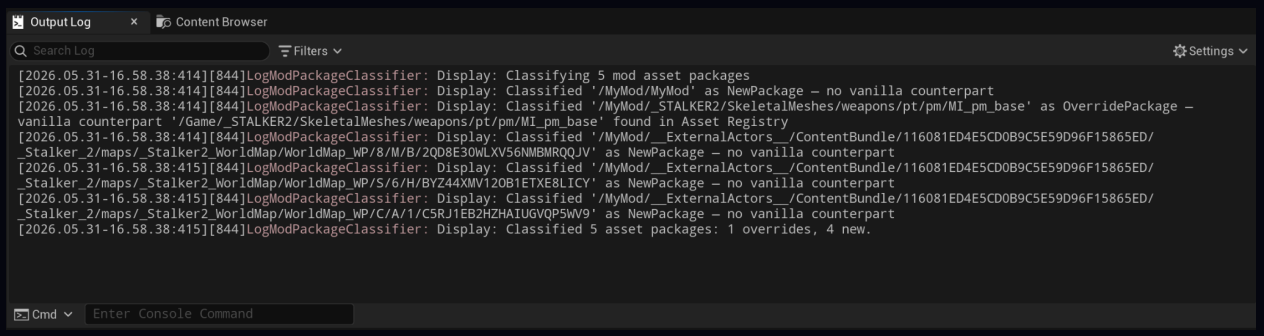
While we're not going to delve into the field differences in this guide, the important property to highlight is the custom Mod field, which indicates both to Zone Kit and to the game that the mod is a game feature and needs to be handled appropriately. This field is required for all mod game features, and without it, they will not be recognized as such.

Packaging

In order to support the new mod functionality, major changes to the mod's cooking and packaging have been made. Let's go through them one by one.

The rework is based on the fact that mods can either modify existing assets of the vanilla game or add new content. And to ensure that both of these aspects work correctly, the packaging pipeline has been split into two: Override Content and New Content. The former mostly matches how legacy .uplugin mods were packaged, while the latter uses a new flow and configuration.

Before any of them are run, an entity called ModPackageClassifier looks through Unreal packages (.uasset files) in the mod's content directory and determines whether a given package is part of the new content or if it's supposed to override an existing package from the vanilla game's content. The results of the classification are printed both to the Output Log window and to corresponding text files, created in the Stalker2/SavedMods/PackageClassifier/<ModName> directory.



The resulting text files of mod package classification are then consumed by the actual mod packaging pipelines. Without delving too deeply into the technicalities, the most important things to understand about the two pipelines are the following:

1. Each pipeline runs a complete Unreal flow: cooking, staging, and packaging. Only actual configurations and parameters are different.
2. Each pipeline produces a complete triplet of output files – .pak, .ucas, and .utoc.
3. The actual mount paths of packaged mod files are different.

Let's look at point 3 in more detail. It is achieved by using a single native parameter when packaging the mods:

```
Engine\Source\Programs\AutomationTool\AutomationUtils\ProjectParams.cs

/// <summary>
/// DLC will remap it's files to the game directory and act like a patch. This is useful if you want to override content in the main game along side your main game.
/// For example having different main game content in different DLCs
/// </summary>
public bool DLCActLikePatch;
```

- The Override Content pipeline sets this parameter to true, ensuring that any processed files are mounted at the Stalker2/Content path when in-game.
- The New Content pipeline sets this parameter to false, ensuring that any processed files are mounted at the Stalker2/Mods/<ModName>/Content path when in-game.

Here's a table with different examples:

File description	Zone Kit file path	Zone Kit package path	Mod package classifier output	Packaged mod file path	Packaged mod package path
A .cfg patch for the vanilla game	Stalker2\Mods\MyMod\Content\GameLite\GameData\WeaponData\CharacterWeaponSettingsPrototypes\MyPlayerWeaponPrototypes.cfg	Not a .uasset file	Not a .uasset file	Stalker2\Content\GameLite\GameData\WeaponData\CharacterWeaponSettingsPrototypes\MyPlayerWeaponPrototypes.cfg	Not a .uasset file
A material override for the vanilla game	Stalker2\Mods\MyMod\Content_STALKER2\SkeletalMeshes\weapons\pt\pm\Ml_pm_base.uasset	/MyMod/_STALKER2/SkeletalMeshes/weapons/pt/pm/Ml_pm_base	Override content	Stalker2\Content_STALKER2\SkeletalMeshes\weapons\pt\pm\Ml_pm_base.uasset	/Game/_STALKER2/SkeletalMeshes/weapons/pt/pm/Ml_pm_base
A new Spawn Actor Prototype	Stalker2\Mods\MyMod\Content\GameLite\ModGameData\SpawnActorPrototypes\WorldMap_WP\1DE858B44C5B84C43D594ABFDEEB2BCB.cfg	Not a .uasset file	Not a .uasset file	Stalker2\Mods\MyMod\Content\GameLite\ModGameData\SpawnActorPrototypes\WorldMap_WP\1DE858B44C5B84C43D594ABFDEEB2BCB.cfg	Not a .uasset file
A new actor using a custom blueprint class	Stalker2\Mods\MyMod\Content\ExternalActors\ContentBundle\116081ED4E5CD0B9C5E59D96F15865ED_Stalker_2\maps_Stalker2_WorldMap\WorldMap_WP\8\M\B\2QD8E30WLXV56NMBMRQQJV.uasset	/MyMod/ExternalActors/ContentBundle/116081ED4E5CD0B9C5E59D96F15865ED/_Stalker_2/maps/_Stalker2_WorldMap/WorldMap_WP/8/M/B/2QD8E30WLXV56NMBMRQQJV	New content	Stalker2\Mods\MyMod\Content\ExternalActors\ContentBundle\116081ED4E5CD0B9C5E59D96F15865ED_Stalker_2\maps_Stalker2_WorldMap\WorldMap_WP\8\M\B\2QD8E30WLXV56NMBMRQQJV.uasset	/MyMod/ExternalActors/ContentBundle/116081ED4E5CD0B9C5E59D96F15865ED/_Stalker_2/maps/_Stalker2_WorldMap/WorldMap_WP/8/M/B/2QD8E30WLXV56NMBMRQQJV