



INTRO TO ACTOR & PLACEHOLDER MODS

Introduction

With the release of Zone Kit Phase 2, mods will be able to modify actors in the game's world. Specifically, you will be able to do the following three main operations:

1. Add new actors.
2. Modify properties of existing actors – patch actors.
3. Delete existing actors.

In the context of S.T.A.L.K.E.R. 2: Heart of Chernobyl, we can subdivide these operations based on the actual actor types:

1. Any **basic** actors.
2. Any **placeholder** actors.

NOTE

From now on, we'll be using the format of actor type modification operation when describing information about a specific method to make it clearer, as most of the actor + modification type combinations are distinct enough, e.g.:

- Placeholder addition – how to add new placeholders.
- Actor patching – how to patch existing basic actors.
- Etc.

Design principle in Zone Kit

Since mods can be used both for modification of the existing vanilla content and addition of new content, each case is handled differently. However, the core idea remains the same everywhere:

1. The team captures the modder's intent, e.g., patching an existing actor.
2. The team generates data based on the captured intent, which later will be processed by the game and/or Zone Kit to directly or indirectly apply that intent, e.g., an actor patch asset is generated, which is processed by the game at runtime.

When adding new content, default Unreal mechanisms apply, e.g.:

- New assets can be created and saved to a disk.
- New actors can be added to the map and saved as external actors.
- Newly added actors can be freely deleted.
- Etc.

However, vanilla assets cannot be directly modified in Zone Kit. While you can freely check out assets, you cannot simply check out existing actors in the game's world. And because of that, any vanilla-modifying mechanisms are handled by separate systems, which will be explained in detail in the respective guides.

Actors vs placeholders

Let's start with simple definitions, most of which you are probably already familiar with:

NOTE

An actor is an entity that can be placed in the game's world. While actors can contain their own properties and logic, most of the time they are used together with components, which are modular and reusable subentities.

Actors are one of the foundational entities in [Unreal's Gameplay Framework](#). In S.T.A.L.K.E.R. 2, there are many different custom actors, but in this guide we'll be specifically looking at placeholders. Placeholders are a part of the simulation system, so let's proceed with the required definitions:

NOTE

A prototype is a set of data, stored in a .cfg file, which defines the initial state of some simulated and/or gameplay-relevant entity. Examples include, but are not limited to, stats of weapons and items, quest and dialog details, AI configurations, etc.

NOTE

A `SpawnActorPrototype` is a set of specific data that defines the initial state of a corresponding actor, which will be spawned using the prototype. Examples include, but are not limited to, loot containers around the world, mutants, NPCs, etc.

NOTE

A placeholder is an **editor-only** actor that generates and updates a corresponding `SpawnActorPrototype` and serves as a visual representation for level designers, quest designers, and modders who work with it.

- Placeholders are immediately destroyed at the start of the game (a PIE session).

- Placeholders are not baked into the game's map.

To highlight the main difference, placeholders are just actors whose sole purpose is to be an **editor-only** representation of a corresponding .cfg file for people to work with.

The definitions have the following implications, so if you are working with placeholders, keep in mind the following:

- Because placeholders are driven by .cfg files, they follow the same .cfg-patching principles and restrictions.
- Because placeholders are still actual actors, which are only present in the editor, the aforementioned actor modification operations can be applied to placeholders as well. But they will obviously be applied only in the editor's world, not in PIE sessions or while in-game.