# Table of Contents

# Save Extension Documentation

Save Extension allows your projects to be saved and loaded.

This plugin is for Unreal Engine 4 and has support for versions **4.20** and **4.19**

# Introduction

At Piperift we like to release the technology we create for ourselves.

Save Extension is part of this technology, and as such, we wanted it to be public so that others can enjoy it too, making the job of the developer considerably easier.

This plugin was designed to fulfill the needs for automatic world saving and loading that are unfortunately missing in the engine at this moment in time. Automatic in the sense that any actor in the world can be saved, including AI, Players, controllers or game logic without any extra components or setups.

# Intended Usage

All our technology is designed to work for very different games and needs, but, naturally, it was created around certain requirements.

In the case of SaveExtension, it has been developed to support games with high amounts of content in the world like open worlds or narrative games.

What I mean by this is that you usually wouldn't serialize a world for a mario game. It can do it, but may not be worth it. Other games might have items to be picked, player states, AI, or streaming levels that require this serialization and here's where the strength of SaveExtension comes.

# Supported Features

#### Asynchronous Saving and Loading

Loading and saving can run asynchronously, splitting the load between frames.
This states can be tracked and shown on UI.

#### Level Streaming and World Composition

Sublevels can be loaded and saved when they get streamed in or out. This allows games to keep the state of the levels even without saving the game.

```
If the player exists an area where 2 enemies were damaged, when he gets in again this enemies will keep their damaged state.
```

#### SaveGame tag saving

Any variable tagged as `SaveGame` will be saved.

#### Full world serialization

All actors in the world are susceptible to be saved.

Only exceptions are for example StaticMeshActors

#### Data Modularity

All data is structured in the way that levels can be loaded at a minimum cost.

#### Compression

Files can be compressed, getting up to 20 times smaller file sizes.

# Asynchronous

# Level Streaming & World Composition

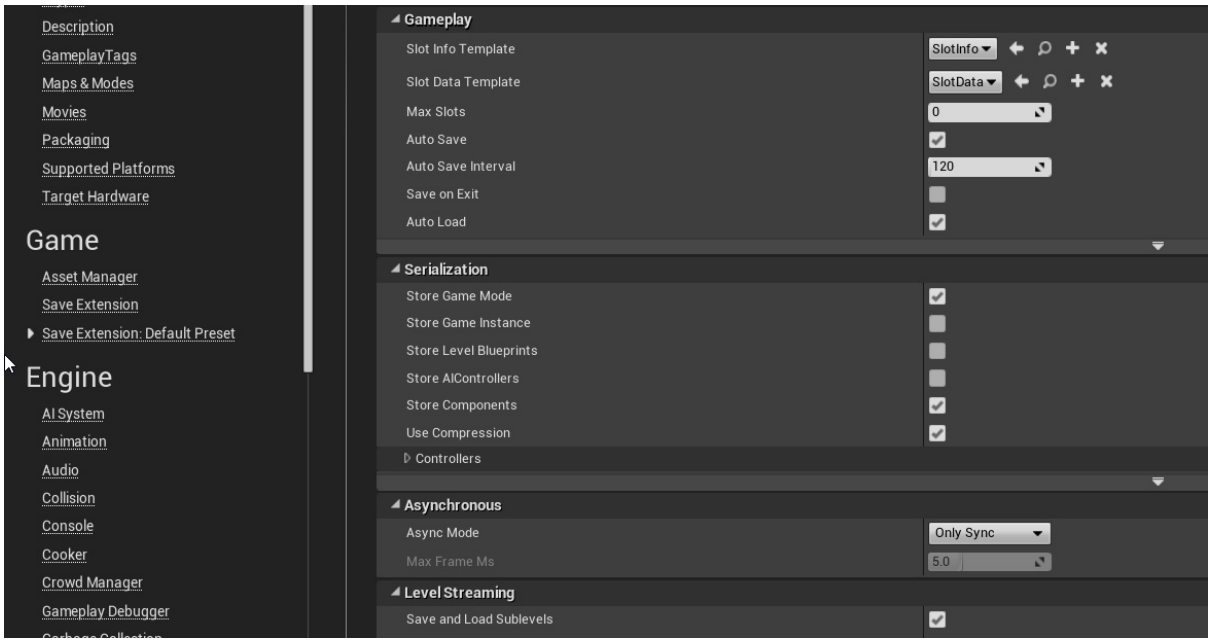
# Presets

**A preset is an asset that serves as a configuration preset for Save Extension.**

Within other settings, presets define how the world is saved, what is saved and what is not.

## Default Preset

Under *Project Settings -> Game -> Save Extension: Default Preset* you will find the default values for all presets.




This default settings page is useful in case you have many presets, or in case you have none (because without any preset, default is used).

> All settings have defined tooltips describing what they are used for

## Gameplay

Defines the runtime behavior of the plugin. Slot templates, maximum numbered slots, autosave, autoload, etc. Debug settings are inside Gameplay as well.

Check Saving & Loading

## Serialization

What should be stored and what should not. Here you can decide if you want to store for example AI.

You can also enable compression to reduce drastically

## Asynchronous

Should load be asynchronous? Should save be asynchronous?

Check Asynchronous

## Level Streaming

Should sublevels be saved and loaded individually?

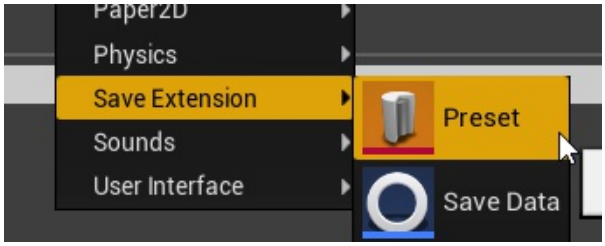
Sublevels will be loaded and saved when they get shown or loaded.

Check Level Streaming

## Multiple presets

Because presets are assets, the active preset can be switched in runtime allowing different saving setups for different maps or gamemodes.

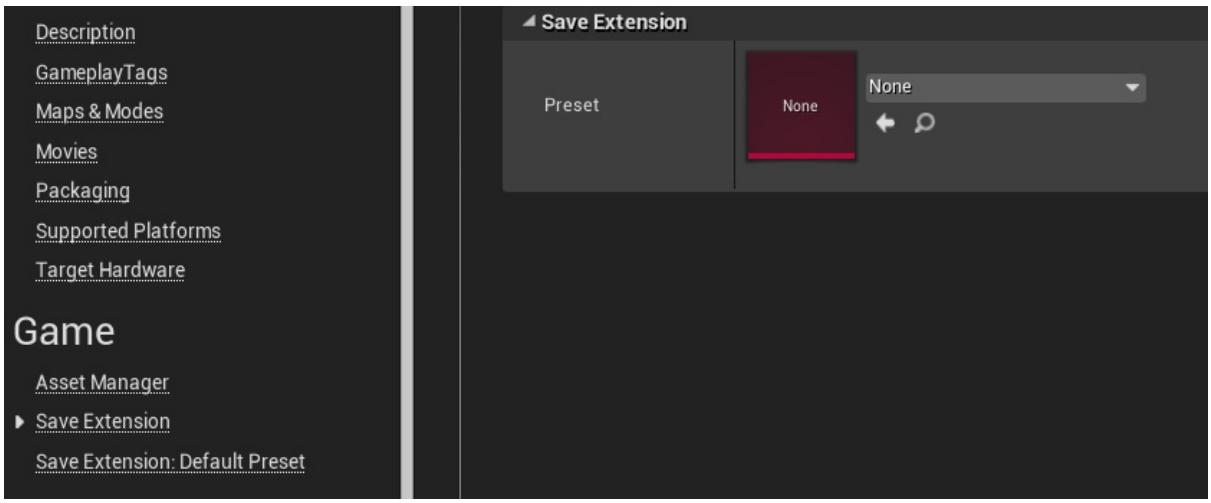
## Creating a Preset




You can create a new preset by right-clicking on the content browser -> *Save Extension* -> *Preset*

## Setting the active Preset

You can set the active preset in editor inside *Project Settings* -> *Game* -> *Save Extension*



# Quick Start

# Saving & Loading

# Slot Templates

# Installation

## Manually

This are the general steps for installing the plugin into your project:

**1.** Download the last release from here
*Make sure you download the same version than your project*

**2.** Extract the folder “SaveExtension” into the **Plugins folder** of your existing project (e.g "MyProject/Plugins")

**2.** Done! You can now open the project

## From Marketplace

Install from the launcher: AVAILABLE HERE