Advanced Save System

Olivier K. Designs

About the plugin:

Advanced Save System is a function library designed for efficient data **saving** and **loading**.

These include:

- **Save Game** (built into **Unreal Engine**)
- **JSON files**
- **Text files**
- **Log files**

This plugin allows **customization** of **file names** and **locations within the project directory** to keep save data organized.

The function library is well-written in **C++**, **fully optimized**, **exposed to Blueprints** and **thoroughly commented**.

Unreal's Save System vs. File-Based Storage:

Unreal Engine's **built-in Save Game System** serializes **UObject-derived classes** into a binary .sav file. This allows for structured data storage that integrates with Unreal's **garbage collection** and **asset system**.

File-based storage methods (**JSON, text files, etc.**) offer more **flexibility** and **readability**, making them ideal for logs, external configs, or cross-platform data transfer. However, they **do not natively support Unreal object references** like the Save Game System does.

Feature	Save Game (.sav)	JSON	Text/Log Files
Binary Format	Yes	No	No
Human-Readable	No	Yes	Yes
Can Store Objects	Yes	No	No
Ideal For	Game saves	Configs, exports	Logs, debug

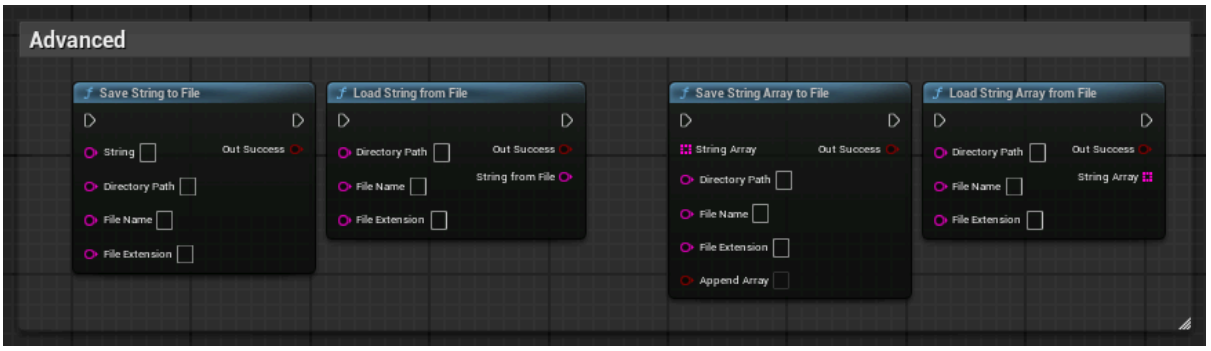
Getting Started:

Advanced Save System is super easy to set up. Since it is **a function library** the functions are always accessible in **Blueprints** and **C++** when including the AdvancedSaveSystem.h in your header file, like so:

```
#include "AdvancedSaveSystem.h"
```

Advanced Features:

If you want **full control** of the data that you'll be saving in **any text-based** form, this is what you'll need to get started



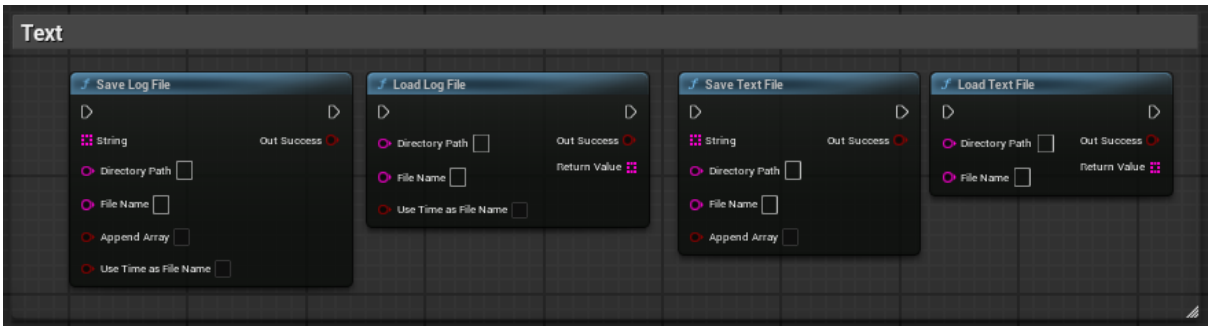
These functions require **more work** than the others, because we wanted to give you **more control** over **saving data in blueprints**.

Let's say you converted a **structure** into **XML** or **JSON** on your own. You can then use these functions to save that and load it by specifying the input **String** or **String Array**, **Directory Path**, **File Name** and **extension**, (e.g, .xml or .JSON.)

When **loading**, the function retrieves the **String** or **String Array** from the **specified Directory Path**, **File Name** and **extension**. Then returns an **Out Success bool** indicating the success.

Text and Log Files:

These are **similar** but use **different file extensions**; we've provided these as the **most basic file save type**.



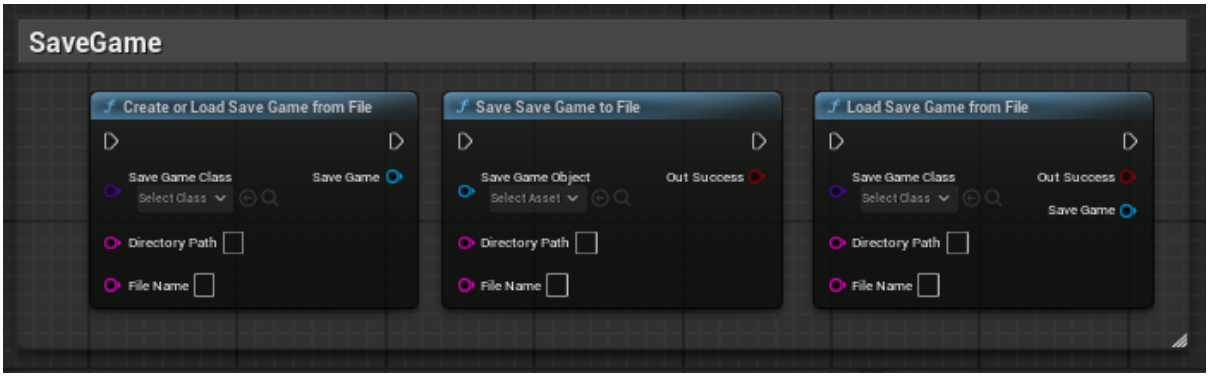
The **input and output** are **similar** to the **advanced functions** with a few additional variables:

- **Append Array Bool** merges the **input Array** with the **existing saved array**.
- **Use Time As File Name Bool** uses the **current time** (Day-Month-Year format) as the **file name** instead of the given **File Name input**.

Save Game System:

Official Documentation: [Unreal Engine Save Games]

In short, **Save Games** store variables and then save them in **binary** (.sav file).



Load Save Game From File function finds the **File** in the **Directory Path** and converts it into a **Save Game**.

- **Out Success Bool** returns **True** if loading was successful.

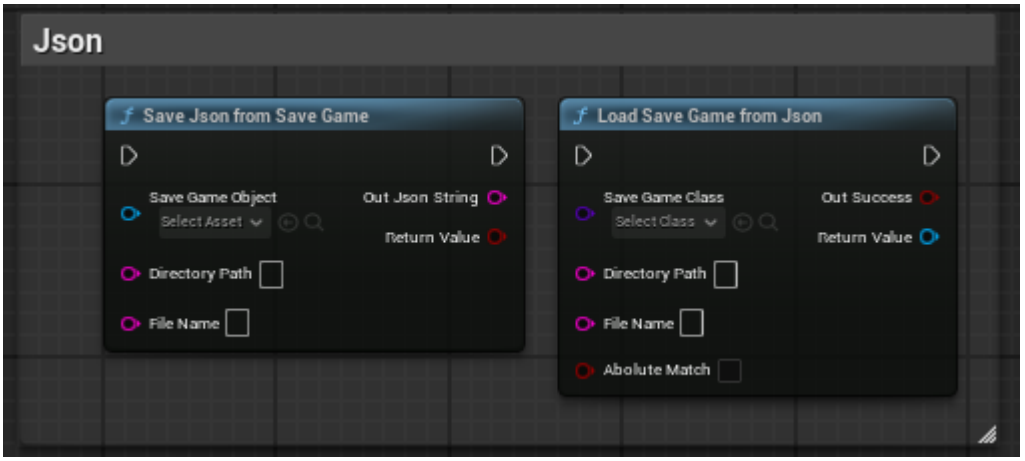
Create Or Load Save Game From File function works like the **Load Save Game From File function**, but **creates** a new **save game** if no file is found.

- **This function always works** because it uses Unreal's built-in **Create Save Game Object function**, which does **not** return a bool.

Save Save Game To File function saves the **Save Game Object** at the **Directory Path** with the **File Name provided**.

- **Out Success Bool Indicates if saving was successful**.

JSON Format:



These will convert a **Save Game** into **JSON** and vice versa and of course they **save** and **load** the **JSON** file respectively.

Save JSON from Save Game,
Requires a **Save Game object**, a **Directory Path**, and a **File Name**.

- **JSON String** converted from the **Save Game**.
- **Out Success Bool** indicating if conversion was **successful**.

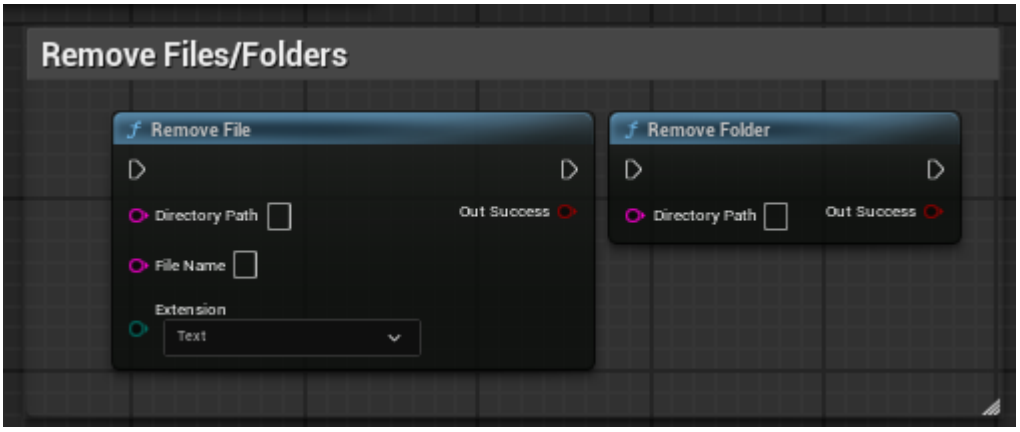
Load Save Game from JSON,
Requires a **Save Game class**, a **Directory Path**, and a **File Name**.

- **Optional Bool Absolute Match** requires all **Save Game** class variables names and type to match **JSON** variables **exactly**; otherwise, returns **false**.
- **Loaded Save Game**.
- **Out Success Bool** indicating if conversion was **successful**.

Important Notes:

- If **Absolute Match is false**, **Out Success** will **always return true**, even if the Save Game **hasn't loaded any values** from the **JSON** file.
- If **Absolute Match is true**, the **Save Game** can be **invalid** if the data does not match correctly.

Utilities:



These functions **remove files** and **folders**.

Remove File function Requires the **Directory Path**, **File Name**, and **Extension**.

Remove Folder function Requires only the **Directory Path** to the **Folder**.