

A Native Serializable Dictionary

Supporting **all** serializable data types



Fully **JSON** Serializable
SerializableDictionary.cs

It is created to be as simple to implement as possible.

Simply follow the standard Dictionary declaration format and apply the [SerializeField] tag to view it in the inspector.

```
[SerializeField]                                JsonConvert.SerializeObject(gameItemsExample);  
2 references  
private SerializableDictionary<string, List<DummyGameItem>> gameItemsExample;
```

For Unity Types which are not normally JSON Serializable, use the SerializableDictionaryBoxed provided. It will allow these value types to be Serialized into JSON.

```
[SerializeField]                                JsonConvert.SerializeObject(example);  
3 references  
private SerializableDictionaryBoxed<Vector3, Color32> example;
```

*All data inputted through the inspector window will be saved automatically. If the data is null or a duplicate key is present, the data will save once the value has been inputted or the duplicate has been resolved. Since it is a fully Native implementation with no editor modifications, there will be no warnings for this as of yet. *Feel free to provide feedback and let me know if you would like a warnings feature implemented.*

What do I mean by *Native*?

`SerializableDictionary` is a Dictionary which does *not* require you to create any classes for different types.

It doesn't use *any* custom editor inspector drawing, any extra or external extensions, or libraries.

It is *fully* native to the Unity C# Environment and displays in the Inspector using *your* Unity version.

It is created as the most lightweight, openly customizable, and extensible `SerializableDictionary` while also including a boxing method for Unity types so they can be **JSON** Serializable.

Supports *All** Data Types


▼ Sample


▼ Keys 2


▼ Warm

Key Warm

▼ Value 3

Element 0 

Element 1 

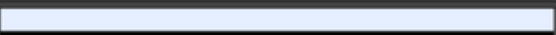
Element 2 


+ -


▼ Cool

Key Cool

▼ Value 3

Element 0 

Element 1 

Element 2 

+ -

+ -

▼ Test 1

▼ Keys 3

▼ Test

Key Test

Value ☐

▼ Test

Key Test

Value ☒

▼ Test

Key Test

Value ☒

+ -

▼ Test 2

▼ Keys 1

▼ Test

Key Test

▼ Value 3

Element 0 ☒

Element 1 ☒

Element 2 ☐

+ -


+ -

▼ Example

▼ Keys 3


▼ Element 0

Key X 1 Key Y 0 Z 9

Value 

▼ Element 1

Key X 1 Key Y 2 Z 0

Value 

► Element 2

+ -

*If you find one that isn't supported, please contact me

+ Classes and Structs

▼ Game Items Example

▼ Keys 3

== ▼ Melee Weapons

Key Melee Weapons

▼ Value 5

== ▼ Dagger

Item UI

Item Name Dagger

Item Image UISprite

Item Description This is a dextrous, lightweight weapon used to swiftly inflict damage to opponents through agile, short motions. This weapon uses slashes and jabs to deliver a combination of strikes inflicting lacerations and delivering blows to vulnerable weakpoints.

Item Stats

Cost 6

Durability 20

Base Damage 4

Drop Chance 84

Stats Roll Range

Item Configuration

Item Type Piercing, Slashing

Crit Multiplier

Swing Pattern

== ▶ Bastard Sword

== ▶ Claymore

== ▶ Mace

== ▶ Halbert

+ -

== ▼ Ranged Weapons

Key Ranged Weapons

▼ Value 4

== ▶ Compound Bow

== ▶ Recurve Bow

== ▶ Flintlock Pistol

== ▶ Blunderbuss

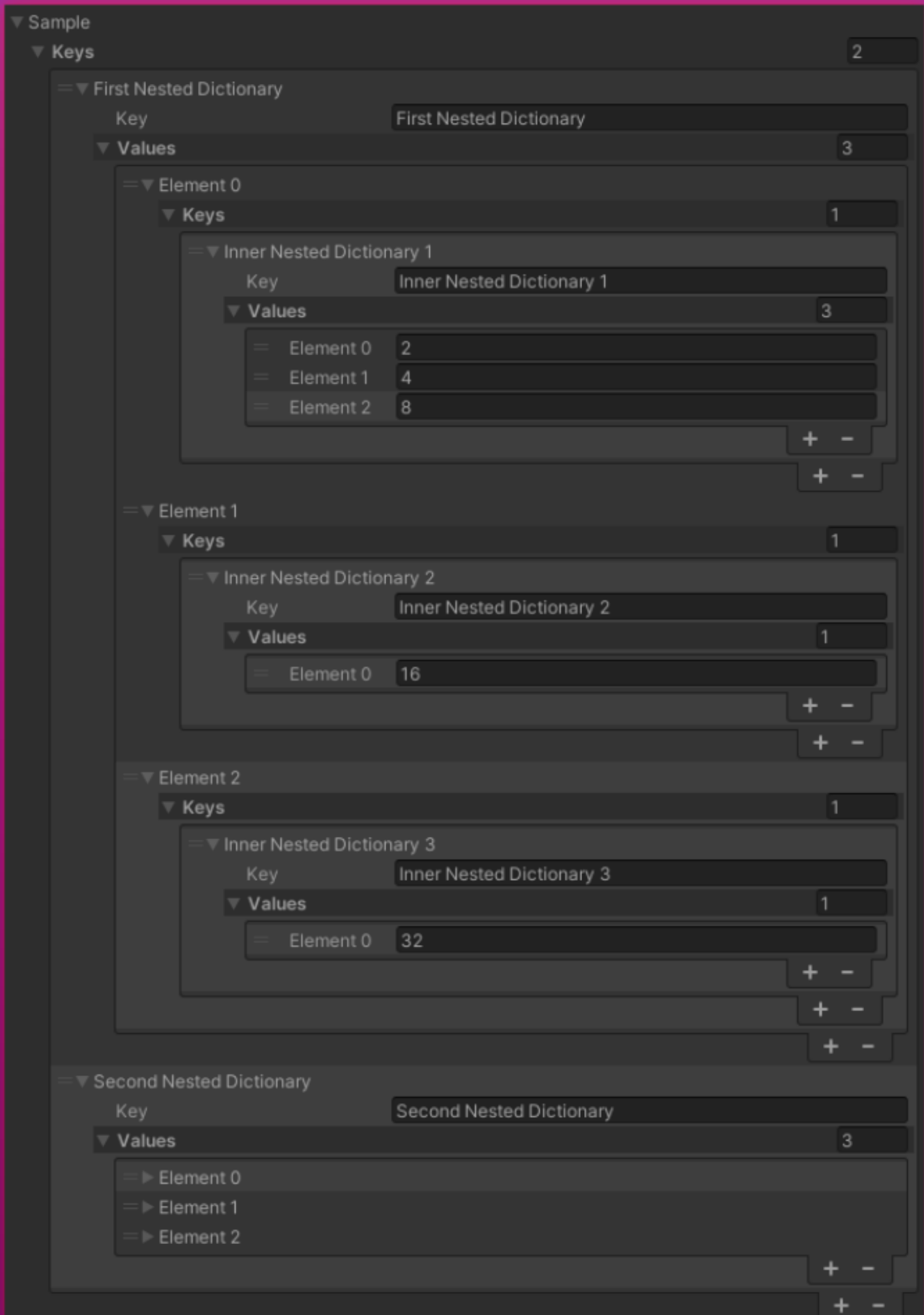
+ -

== ▼ Other Weapons

Key Other Weapons

You can also bind ScriptableObject's data to it!

Supports Complex Data



The Nesting capabilities are only limited by Unity's internal draw limits