Kantan Auto-JSON User Guide

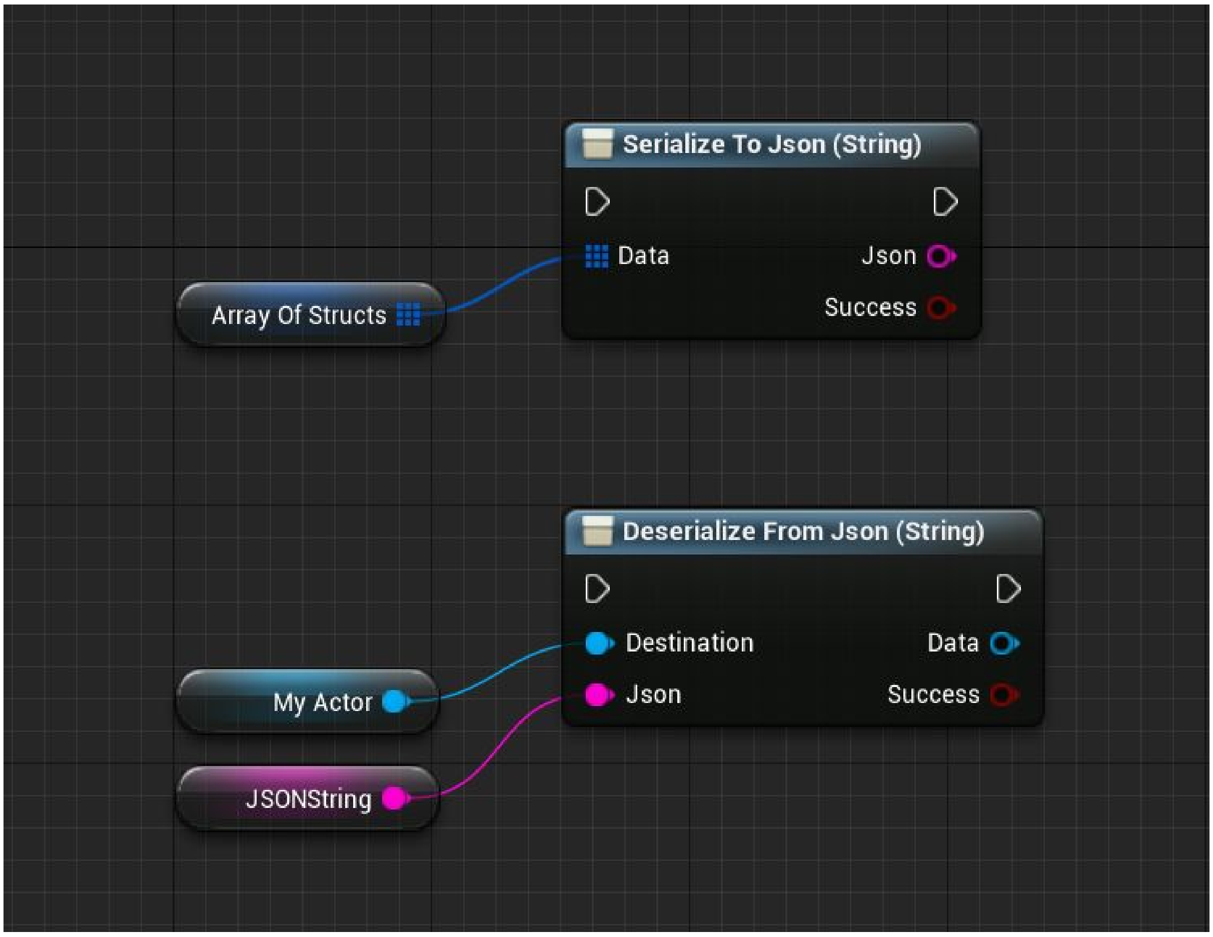
This is a very simple to use plugin that allows for single-node automatic conversion between UE4 blueprint types (objects, structs, arrays, sets and maps) and JSON text.

Getting Started

Once the plugin is installed, you should be able to use the provided nodes in any blueprint. To check it's installed and active, open the *Plugins* tab, and look for *Kantan AutoJSON* under the category *Kantan Dev*.

Basic Usage Examples

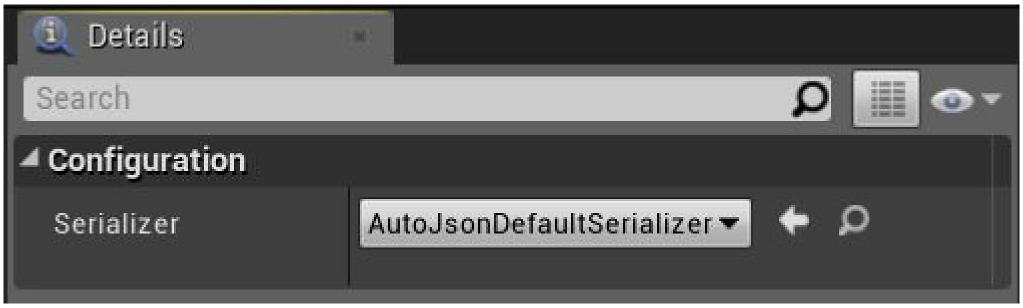
The plugin provides custom nodes which make use of wildcard pins. This way, the same node can be used regardless of the structure of the data you want to convert. The example below shows plugging in an array of structs to convert to JSON, and also extracting actor properties from previously saved JSON.



On the *Deserialize* node, the *Data* output pin is just for convenience, it provides a reference to whatever was passed in as the destination for the conversion.

Serializer Types

Selecting a Serialize or Deserialize node in a blueprint will show its properties in the *Details* panel.



Serializer: AutoJsonDefaultSerializer

The default serializer will convert all properties on a struct or object, except for those marked as *Transient* (this flag can be set for properties in C++, and for variables in blueprints in the expanded dropdown).

There is also a SaveGame serializer provided by the plugin. This is not specifically for saved game scenarios - it simply serializes only those properties that are marked with the *SaveGame* flag.

It is also possible to create your own custom serializers in C++ if more control is needed.

Note that you should use the same serializer when loading data that you used when saving it.

Conversion Rules

The mapping between UE4 and JSON is very straightforward.

UE4 Type	JSON Value Type	Notes
Integer, Byte, Float	Number	
Boolean	true/false	
Name, String, Text	String	
Array, Set	Array	
Map	Object	Map key type is restricted to Name or String. Each entry in the map will correspond to a name-value pair in the JSON object.
Struct, Object	Object	Each serialized property in the UE4 struct/object will be an entry in the JSON object, with the name part being the name of the property. For example, a *Vector* will serialize as: `{ “X”: [value], “Y”: [value], “Z”: [value] }`

A Note on UE4 Objects

The plugin does not support spawning new objects or actors as part of the deserialization process. Deserializing an object requires you to have an existing object instance. You pass this into the *Deserialize* node, and that object’s properties will then be populated with the data read from the JSON.