# 298.https://stackoverflow.com/questions/71740417/how-do-i-create-a-list-or-array-of-different-data-types-from-c-sharp-to-be-saved

**T:**How do I create a List or Array of different data types from C# to be saved to JSON in Unity

**Q:**I would like my output JSON to contain a simple array shown below  
  
{"attributes":[ { "trait\_type": "Background", "value": "Green" }, { "trait\_type": "Body", "value": "Body\_1" }, { "trait\_type": "Outfit", "value": "Beach\_Singlet" }, { "display\_type":"date", "trait\_type":"birthday", "value":869270400 } ]}  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]   
  
Notice how the last item is different from the previous items in the array. The variable named "value" is also an integer as compared to the previous entries as strings.  
  
How do I go about in order to be able to output my JSON as shown above? I have tried creating a class that can store all the information, but I cannot reuse the name "value" for both an int and string declaration, and also do not wish to show the variables if their value is null(Example shown below)  
  
{ "attributes": [ { "display\_type": "", "trait\_type": "Background", "value": "Green" }, { "display\_type": "", "trait\_type": "Body", "value": "Body\_1" }, { "display\_type": "", "trait\_type": "Outfit", "value": "Beach\_Singlet" }, { "display\_type": "date", "trait\_type": "birthday", "value": 869270400 } ]}  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]

**C1:**Do you absolutely have to do this? It's always a pain in the rear end when someone's designed their API to do stuff thus

**C2:**I am following the standard NFT metadata json convention, or am I missing anything here? This format is so that Opensea can properly read my JSON and output the way I want it to be. docs.opensea.io/docs/metadata-standards

**C3:**@Martin I can not see any problem to create a list of string and int as a string.

2 **Answer**

**A1:**You can use object type.  
  
using Newtonsoft.Json;var list = new AttributeList { attributes = new []{ new Attribute { trait\_type = "Background", value = "green" }, new Attribute { display\_type = "date", trait\_type = "birthday", value = 869270400 } } };var json = JsonConvert.SerializeObject(list, Formatting.Indented);Console.WriteLine(json);public class Attribute{ public object value { get; set; } public string trait\_type { get; set; } [JsonProperty(NullValueHandling = NullValueHandling.Ignore)] public string display\_type { get; set; }}public class AttributeList{ public Attribute[] attributes { get; set; }}  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]   
  
Output:  
  
 { "attributes": [ { "value": "green", "trait\_type": "Background" }, { "value": 869270400, "trait\_type": "birthday", "display\_type": "date" } ]}  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]

**C1:**This is very easy to understand, thanks a lot!

**A2:**try this  
  
 var attributes=new List<Attribute>{ new AttributeString{ trait\_type="Background", value="green" }, new AttributeInt{ display\_type ="date", trait\_type="birthday", value=869270400 } }; var jsonSerializerSettings = new JsonSerializerSettings() { TypeNameHandling = TypeNameHandling.Objects, NullValueHandling=NullValueHandling.Ignore, Formatting=Newtonsoft.Json.Formatting.Indented }; var json = JsonConvert.SerializeObject(attributes,jsonSerializerSettings);  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]   
  
classes  
  
public class Attribute{ public string trait\_type { get; set; } public string display\_type { get; set; }}public class AttributeString:Attribute{ public string value { get; set; }}public class AttributeInt:Attribute{ public int value { get; set; }}public class AttributeList{ public List<Attribute> attributes { get; set; }}  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]