|  |
| --- |
| Serialization using ContractResolver |



The [IContractResolver](https://www.newtonsoft.com/json/help/html/T_Newtonsoft_Json_Serialization_IContractResolver.htm) interface provides a way to customize how the JsonSerializer serializes and deserializes .NET objects to JSON without placing attributes on your classes.

Anything that can be set on an object, collection, property, etc, using attributes or methods to control serialization can also be set using an IContractResolver.

* [DefaultContractResolver](https://www.newtonsoft.com/json/help/html/ContractResolver.htm#DefaultContractResolver)
* [CamelCasePropertyNamesContractResolver](https://www.newtonsoft.com/json/help/html/ContractResolver.htm#CamelCasePropertyNamesContractResolver)
* [Custom IContractResolver Examples](https://www.newtonsoft.com/json/help/html/ContractResolver.htm#CustomIContractResolverExamples)

https://www.newtonsoft.com/json/help/icons/SectionExpanded.pngDefaultContractResolver

The [DefaultContractResolver](https://www.newtonsoft.com/json/help/html/T_Newtonsoft_Json_Serialization_DefaultContractResolver.htm) is the default resolver used by the serializer. It provides many avenues of extensibility in the form of virtual methods that can be overridden.

https://www.newtonsoft.com/json/help/icons/SectionExpanded.pngCamelCasePropertyNamesContractResolver

[CamelCasePropertyNamesContractResolver](https://www.newtonsoft.com/json/help/html/T_Newtonsoft_Json_Serialization_CamelCasePropertyNamesContractResolver.htm) inherits from DefaultContractResolver and simply overrides the JSON property name to be written in [camelcase](http://en.wikipedia.org/wiki/CamelCase).

**ContractResolver**

[Copy](https://www.newtonsoft.com/json/help/html/ContractResolver.htm)

Product product = new Product

{

ExpiryDate = new DateTime(2010, 12, 20, 18, 1, 0, DateTimeKind.Utc),

Name = "Widget",

Price = 9.99m,

Sizes = new[] { "Small", "Medium", "Large" }

};

string json =

JsonConvert.SerializeObject(

product,

Formatting.Indented,

new JsonSerializerSettings { ContractResolver = new CamelCasePropertyNamesContractResolver() }

);

//{

// "name": "Widget",

// "expiryDate": "2010-12-20T18:01:00Z",

// "price": 9.99,

// "sizes": [

// "Small",

// "Medium",

// "Large"

// ]

//}

https://www.newtonsoft.com/json/help/icons/SectionExpanded.pngCustom IContractResolver Examples

**Use JsonConverter with IContractResolver**

[Copy](https://www.newtonsoft.com/json/help/html/ContractResolver.htm)

public class ConverterContractResolver : DefaultContractResolver

{

public new static readonly ConverterContractResolver Instance = new ConverterContractResolver();

protected override JsonContract CreateContract(Type objectType)

{

JsonContract contract = base.CreateContract(objectType);

// this will only be called once and then cached

if (objectType == typeof(DateTime) || objectType == typeof(DateTimeOffset))

{

contract.Converter = new JavaScriptDateTimeConverter();

}

return contract;

}

}

This example sets a [JsonConverter](https://www.newtonsoft.com/json/help/html/T_Newtonsoft_Json_JsonConverter.htm) for a type using an IContractResolver. Using a contract resolver here is useful because DateTime is not your own type and it is not possible to place a JsonConverterAttribute on it.

**Conditional properties with IContractResolver**

[Copy](https://www.newtonsoft.com/json/help/html/ContractResolver.htm)

public class ShouldSerializeContractResolver : DefaultContractResolver

{

public new static readonly ShouldSerializeContractResolver Instance = new ShouldSerializeContractResolver();

protected override JsonProperty CreateProperty(MemberInfo member, MemberSerialization memberSerialization)

{

JsonProperty property = base.CreateProperty(member, memberSerialization);

if (property.DeclaringType == typeof(Employee) && property.PropertyName == "Manager")

{

property.ShouldSerialize =

instance =>

{

Employee e = (Employee)instance;

return e.Manager != e;

};

}

return property;

}

}

This example sets up [conditional serialization for a property](https://www.newtonsoft.com/json/help/html/ConditionalProperties.htm) using an IContractResolver. This is useful if you want to conditionally serialize a property but don't want to add additional methods to your type.