

Working with Files



Filip Ekberg

Principal Consultant & CEO

@fekberg | fekberg.com



**Serializing and Deserializing
types that rely on inheritance
may not work in older versions
of System.Text.Json without
lots of additional code.**



TypeNameHandling

- “None** **= Do not include the .NET type name when serializing types.**
- Objects** **= Include the .NET type name when serializing into a JSON object structure.**
- Arrays** **= Include the .NET type name when serializing into a JSON array structure.**
- All** **= Always include the .NET type name when serializing.**
- Auto** **= Include the .NET type name when the type of the object being serialized is not the same as its declared type.**
Note that this doesn't include the root serialized object by default.”



**Avoid ambiguity by
following the standard
ISO 8601!**



Using NodaTime

```
using NodaTime; // Install-Package NodaTime

var now = SystemClock.Instance
    .GetCurrentInstant();

var stockholmTimeZone =
    DateTimeZoneProviders.Tzdb[ "Europe/Stockholm" ];

var swedenTime = now.InZone(stockholm);
```



Avoid CurrentCulture and use Invariant Culture

```
int number          = 1;  
decimal number2     = 150.5m;  
decimal number3     = -1_500_000.1337m;
```

```
number.ToString();
```

```
number2.ToString();
```

```
number3.ToString();
```

Will use the **CurrentCulture** to determine how to represent the string.

NOT GOOD when persisting data!



Use Invariant Culture to Store and Restore Data

```
// Prepare data to be stored
string data = 1_500_000.50.ToString(CultureInfo.InvariantCulture);

// Restore data
var number = decimal.Parse(data, CultureInfo.InvariantCulture);
```



**JSON.NET uses
InvariantCulture by default
which is the best practice
when serializing data**



**Importing and Exporting
data may seem trivial but it
is not always!**



Example: ISO 8601

Year Month Day Hour Minute Second

↓ ↓ ↓ ↓ ↓ ↓

2021-05-10T19:30:00+00:00

↑ ↑

Time delimiter Time zone offset or Zulu time (Z)

The diagram illustrates the components of the ISO 8601 timestamp '2021-05-10T19:30:00+00:00'. Above the string, labels identify each part: 'Year' for '2021', 'Month' for '05', 'Day' for '10', 'Hour' for '19', 'Minute' for '30', and 'Second' for '00'. Pink arrows point from these labels to their respective digits. Below the string, two more labels are present: 'Time delimiter' with an arrow pointing to the 'T' character, and 'Time zone offset or Zulu time (Z)' with an arrow pointing to the '+00:00' suffix.



Example: Decimal Separator

```
var price = decimal.Parse("10,1");
```



sv-SE = 10 dollars and 10 cents
en-US = 101 dollars



Some of the code in the UI project can be extracted to a separate layer – as it is not UI specific and could be shared!

