

# System.Byte Structure

```
[ILAsm]
.class public sequential sealed serializable Byte extends
System.ValueType implements System.IComparable, System.IFormattable,
System.IComparable`1<unsigned int8>, System.IEquatable`1<unsigned
int8>

[C#]
public struct Byte: IComparable, IFormattable, IComparable<Byte>,
IEquatable<Byte>
```

## Assembly Info:

- *Name:* mscorlib
- *Public Key:* [00 00 00 00 00 00 00 00 04 00 00 00 00 00 00 00]
- *Version:* 2.0.x.x
- *Attributes:*
  - CLSCompliantAttribute(true)

## Implements:

- **System.IComparable**
- **System.IFormattable**
- **System.IComparable<System.Byte>**
- **System.IEquatable<System.Byte>**

## Summary

Represents an 8-bit unsigned integer.

## Inherits From: System.ValueType

**Library:** BCL

**Thread Safety:** This type is safe for multithreaded operations.

## Description

The `System.Byte` data type represents integer values ranging from 0 to positive 255 (hexadecimal 0xFF).

# Byte.MaxValue Field

```
[ILAsm]  
.field public static literal unsigned int8 MaxValue = 255
```

```
[C#]  
public const byte MaxValue = 255
```

## Summary

Contains the maximum value for the `System.Byte` type.

## Description

The value of this constant is 255 (hexadecimal 0xFF).

# Byte.MinValue Field

```
[ILAsm]  
.field public static literal unsigned int8 MinValue = 0
```

```
[C#]  
public const byte MinValue = 0
```

## Summary

Contains the minimum value for the `System.Byte` type.

## Description

The value of this constant is 0.

# Byte.CompareTo(System.Byte) Method

```
[ILAsm]  
.method public final hidebysig virtual int32 CompareTo(unsigned int8  
value)
```

```
[C#]  
public int CompareTo(byte value)
```

## Summary

Returns the sort order of the current instance compared to the specified unsigned byte.

## Parameters

Parameter	Description
<i>value</i>	The <code>System.Byte</code> to compare to the current instance.

## Return Value

The return value is a negative number, zero, or a positive number reflecting the sort order of the current instance as compared to *value*. For non-zero return values, the exact value returned by this method is unspecified. The following table defines the return value:

Return Value	Description
A negative number	Current instance < <i>value</i> .
Zero	Current instance == <i>value</i> .
A positive number	Current instance > <i>value</i> .

## Description

[*Note:* This method is implemented to support the `System.IComparable<Byte>` interface.]

# Byte.CompareTo(System.Object) Method

```
[ILAsm]  
.method public final hidebysig virtual int32 CompareTo(object value)
```

```
[C#]  
public int CompareTo(object value)
```

## Summary

Returns the sort order of the current instance compared to the specified object.

## Parameters

Parameter	Description
<i>value</i>	The <i>System.Object</i> to compare to the current instance.

## Return Value

The return value is a negative number, zero, or a positive number reflecting the sort order of the current instance as compared to *value*. For non-zero return values, the exact value returned by this method is unspecified. The following table defines the return value:

Return Value	Description
A negative number	Current instance < <i>value</i> .
Zero	Current instance == <i>value</i> .
A positive number	Current instance > <i>value</i> , or <i>value</i> is a null reference.

## Description

[*Note:* This method is implemented to support the *System.IComparable* interface.]

## Exceptions

Exception	Condition
<b>System.ArgumentException</b>	<i>value</i> is not a <i>System.Byte</i> and is not a null reference.



# Byte.Equals(System.Byte) Method

```
[ILAsm]  
.method public hidebysig virtual bool Equals(unsigned int8 obj)  
  
[C#]  
public override bool Equals(byte obj)
```

## Summary

Determines whether the current instance and the specified `System.Byte` represent the same value.

## Parameters

Parameter	Description
<i>obj</i>	The <code>System.Object</code> to compare to the current instance.

## Return Value

`true` if *obj* represents the same value as the current instance; otherwise, `false`.

## Description

[*Note:* This method is implemented to support the `System.IEquatable<Byte>` interface.]

# Byte.Equals(System.Object) Method

```
[ILAsm]  
.method public hidebysig virtual bool Equals(object obj)
```

```
[C#]  
public override bool Equals(object obj)
```

## Summary

Determines whether the current instance and the specified `System.Object` represent the same type and value.

## Parameters

Parameter	Description
<i>obj</i>	The <code>System.Object</code> to compare to the current instance.

## Return Value

`true` if *obj* represents the same type and value as the current instance. If *obj* is a null reference or is not an instance of `System.Byte`, returns `false`.

## Description

[*Note:* This method overrides `System.Object.Equals`.]



# Byte.GetHashCode() Method

```
[ILAsm]  
.method public hidebysig virtual int32 GetHashCode()  
  
[C#]  
public override int GetHashCode()
```

## Summary

Generates a hash code for the current instance.

## Return Value

A `System.Int32` containing the hash code for the current instance.

## Description

The algorithm used to generate the hash code is unspecified.

[*Note:* This method overrides `System.Object.GetHashCode.`]

# Byte.Parse(System.String) Method

```
[ILAsm]  
.method public hidebysig static unsigned int8 Parse(string s)  
  
[C#]  
public static byte Parse(string s)
```

## Summary

Returns the specified `System.String` converted to a `System.Byte` value.

## Parameters

Parameter	Description
<code>s</code>	A <code>System.String</code> containing the value to convert. The string is interpreted using the <code>System.Globalization.NumberStyles.Integer</code> style.

## Return Value

The `System.Byte` value obtained from `s`.

## Description

This version of `System.Byte.Parse` is equivalent to `System.Byte.Parse (s, System.Globalization.NumberStyles.Integer, null )`.

The string `s` is parsed using the formatting information in a `System.Globalization.NumberFormatInfo` initialized for the current system culture. [*Note:* For more information, see `System.Globalization.NumberFormatInfo.CurrentInfo`.]

## Exceptions

Exception	Condition
<b>System.ArgumentNullException</b>	<code>s</code> is a null reference.
<b>System.FormatException</b>	<code>s</code> is not in the correct style.
<b>System.OverflowException</b>	<code>s</code> represents a number greater than <code>System.Byte.MaxValue</code> or less than <code>System.Byte.MinValue</code> .

## Example

The following example demonstrates the `System.Byte.Parse` method.

[C#]

```
using System;
public class ByteParseClass {
public static void Main() {
    string str = " 100 ";
    Console.WriteLine("String: \"{0}\" <Byte> {1}",str,Byte.Parse(str));
}
}
```

The output is

```
String: " 100 " <Byte> 100
```

# Byte.Parse(System.String, System.Globalization.NumberStyles) Method

```
[ILAsm]  
.method public hidebysig static unsigned int8 Parse(string s,  
valuetype System.Globalization.NumberStyles style)
```

```
[C#]  
public static byte Parse(string s, NumberStyles style)
```

## Summary

Returns the specified `System.String` converted to a `System.Byte` value.

## Parameters

Parameter	Description
<i>s</i>	A <code>System.String</code> containing the value to convert. The string is interpreted using the style specified by <i>style</i> .
<i>style</i>	Zero or more <code>System.Globalization.NumberStyles</code> values that specify the style of <i>s</i> . Specify multiple values for <i>style</i> using the bitwise OR operator. If <i>style</i> is a null reference, the string is interpreted using the <code>System.Globalization.NumberStyles.Integer</code> style.

## Return Value

The `System.Byte` value obtained from *s*.

## Description

This version of `System.Byte.Parse` is equivalent to `System.Byte.Parse(s, style, null)`.

The string *s* is parsed using the formatting information in a `System.Globalization.NumberFormatInfo` initialized for the current system culture. [*Note:* For more information, see `System.Globalization.NumberFormatInfo.CurrentInfo`.]

## Exceptions

Exception	Condition
-----------	-----------

<b>System.ArgumentNullException</b>	s is a null reference.
<b>System.FormatException</b>	s is not in the correct style.
<b>System.OverflowException</b>	s represents a number greater than <code>System.Byte.MaxValue</code> or less than <code>System.Byte.MinValue</code> .

# Byte.Parse(System.String, System.IFormatProvider) Method

```
[ILAsm]  
.method public hidebysig static unsigned int8 Parse(string s, class  
System.IFormatProvider provider)  
  
[C#]  
public static byte Parse(string s, IFormatProvider provider)
```

## Summary

Returns the specified `System.String` converted to a `System.Byte` value.

## Parameters

Parameter	Description
<i>s</i>	A <code>System.String</code> containing the value to convert. The string is interpreted using the <code>System.Globalization.NumberStyles.Integer</code> style.
<i>provider</i>	A <code>System.IFormatProvider</code> that supplies a <code>System.Globalization.NumberFormatInfo</code> containing culture-specific formatting information about <i>s</i> .

## Return Value

The `System.Byte` value obtained from *s*.

## Description

This version of `System.Byte.Parse` is equivalent to `System.Byte.Parse(s, System.Globalization.NumberStyles.Integer, provider)`.

The string *s* is parsed using the culture-specific formatting information from the `System.Globalization.NumberFormatInfo` instance supplied by *provider*. If *provider* is null or a `System.Globalization.NumberFormatInfo` cannot be obtained from *provider*, the formatting information for the current system culture is used.

## Exceptions

Exception	Condition
<code>System.ArgumentNullException</code>	<i>s</i> is a null reference.

<b>System.OverflowException</b>	s represents a number greater than <code>System.Byte.MaxValue</code> or less than <code>System.Byte.MinValue</code> .
<b>System.FormatException</b>	s is not in the correct style.

# Byte.Parse(System.String, System.Globalization.NumberStyles, System.IFormatProvider) Method

```
[ILAsm]  
.method public hidebysig static unsigned int8 Parse(string s,  
valuetype System.Globalization.NumberStyles style, class  
System.IFormatProvider provider)
```

```
[C#]  
public static byte Parse(string s, NumberStyles style,  
IFormatProvider provider)
```

## Summary

Returns the specified `System.String` converted to a `System.Byte` value.

## Parameters

Parameter	Description
<i>s</i>	A <code>System.String</code> containing the value to convert. The string is interpreted using the style specified by <i>style</i> .
<i>style</i>	Zero or more <code>System.Globalization.NumberStyles</code> values that specify the style of <i>s</i> . Specify multiple values for <i>style</i> using the bitwise OR operator. If <i>style</i> is a null reference, the string is interpreted using the <code>System.Globalization.NumberStyles.Integer</code> style.
<i>provider</i>	A <code>System.IFormatProvider</code> that supplies a <code>System.Globalization.NumberFormatInfo</code> containing culture-specific formatting information about <i>s</i> .

## Return Value

The `System.Byte` value obtained from *s*.

## Description

The string *s* is parsed using the culture-specific formatting information from the `System.Globalization.NumberFormatInfo` instance supplied by *provider*. If *provider* is null or a `System.Globalization.NumberFormatInfo` cannot be obtained from *provider*, the formatting information for the current system culture is used.

## Exceptions



Exception	Condition
<b>System.ArgumentNullException</b>	s is a null reference.
<b>System.FormatException</b>	s is not in the correct style.
<b>System.OverflowException</b>	s represents a number greater than System.Byte.MaxValue or less than System.Byte.MinValue.

# Byte.ToString(System.IFormatProvider) Method

```
[ILAsm]  
.method public final hidebysig virtual string ToString(class  
System.IFormatProvider provider)  
  
[C#]  
public string ToString(IFormatProvider provider)
```

## Summary

Returns a `System.String` representation of the value of the current instance.

## Parameters

Parameter	Description
<i>provider</i>	A <code>System.IFormatProvider</code> that supplies a <code>System.Globalization.NumberFormatInfo</code> containing culture-specific formatting information.

## Return Value

A `System.String` representation of the current instance formatted using the general format specifier, ("G"). The string takes into account the information in the `System.Globalization.NumberFormatInfo` instance supplied by *provider*.

## Description

This version of `System.Byte.ToString` is equivalent to `System.Byte.ToString("G", provider)`.

If *provider* is null or a `System.Globalization.NumberFormatInfo` cannot be obtained from *provider*, the formatting information for the current system culture is used.

# Byte.ToString(System.String, System.IFormatProvider) Method

```
[ILAsm]  
.method public final hidebysig virtual string ToString(string  
format, class System.IFormatProvider provider)  
  
[C#]  
public string ToString(string format, IFormatProvider provider)
```

## Summary

Returns a `System.String` representation of the value of the current instance.

## Parameters

Parameter	Description
<i>format</i>	A <code>System.String</code> containing a character that specifies the format of the returned string.
<i>provider</i>	A <code>System.IFormatProvider</code> that supplies a <code>System.Globalization.NumberFormatInfo</code> instance containing culture-specific formatting information.

## Return Value

A `System.String` representation of the current instance formatted as specified by *format*. The string takes into account the information in the `System.Globalization.NumberFormatInfo` instance supplied by *provider*.

## Description

If *provider* is `null` or a `System.Globalization.NumberFormatInfo` cannot be obtained from *provider*, the formatting information for the current system culture is used.

If *format* is a `null` reference, the general format specifier "G" is used.

The following table lists the characters that are valid for the `System.Byte` type:

Format Characters	Description
"C", "c"	Currency format.
"D", "d"	Decimal format.
"E", "e"	Exponential notation format.
"F", "f"	Fixed-point format.

"G", "g"	General format.
"N", "n"	Number format.
"P", "p"	Percent format.
"X", "x"	Hexadecimal format.

[*Note:* For a detailed description of formatting, see the `System.IFormattable` interface.

This method is implemented to support the `System.IFormattable` interface.

]

## Exceptions

Exception	Condition
<b>System.FormatException</b>	<i>format</i> is invalid.

# Byte.ToString() Method

```
[ILAsm]  
.method public hidebysig virtual string ToString()  
  
[C#]  
public override string ToString()
```

## Summary

Returns a `System.String` representation of the value of the current instance.

## Return Value

A `System.String` representation of the current instance formatted using the general format specifier ("G"). The string takes into account the current system culture.

## Description

This version of `System.Byte.ToString` is equivalent to `System.Byte.ToString (null, null )`.

[*Note:* This method overrides `System.Object.ToString`.]

# Byte.ToString(System.String) Method

```
[ILAsm]  
.method public hidebysig instance string ToString(string format)
```

```
[C#]  
public string ToString(string format)
```

## Summary

Returns a `System.String` representation of the value of the current instance.

## Parameters

Parameter	Description
<i>format</i>	A <code>System.String</code> that specifies the format of the returned string. [Note: For a list of valid values, see <code>System.Byte.ToString(System.String, System.IFormatProvider)</code> .]

## Return Value

A `System.String` representation of the current instance formatted as specified by *format*. The string takes into account the current system culture.

## Description

This version of `System.Byte.ToString` is equivalent to `System.Byte.ToString(format, null)`.

If *format* is `null`, the general format specifier "G" is used.

## Exceptions

Exception	Condition
<code>System.FormatException</code>	<i>format</i> is invalid.

## Example

The following example demonstrates the `System.Byte.ToString` method.

```
[C#]
```

```
using System;  
public class ByteToStringExample {
```

```
public static void Main() {  
    Byte b = 8;  
    Console.WriteLine(b);  
    String[] formats = {"c", "d", "e", "f", "g", "n", "p", "x" };  
    foreach(String str in formats)  
        Console.WriteLine("{0}: {1}", str, b.ToString(str));  
}
```

The output is

8

c: \$8.00

d: 8

e: 8.000000e+000

f: 8.00

g: 8

n: 8.00

p: 800.00 %

x: 8