

# 1 System.UInt32 Structure

```
2 [ILAsm]  
3 .class public sequential sealed serializable UInt32 extends  
4 System.ValueType implements System.IComparable, System.IFormattable,  
5 System.IComparable`1<unsigned int32>, System.IEquatable`1<unsigned int32>  
  
6 [C#]  
7 public struct UInt32: IComparable, IFormattable, IComparable<UInt32>,  
8 IEquatable<UInt32>
```

## 9 Assembly Info:

- 10 • *Name:* mscorlib
- 11 • *Public Key:* [00 00 00 00 00 00 00 00 04 00 00 00 00 00 00]
- 12 • *Version:* 2.0.x.x
- 13 • *Attributes:*
  - 14 ○ CLSCompliantAttribute(true)

## 15 Type Attributes:

- 16 • CLSCompliantAttribute(false)

## 17 Implements:

- 18 • **System.IComparable**
- 19 • **System.IFormattable**
- 20 • **System.IComparable<System.UInt32>**
- 21 • **System.IEquatable<System.UInt32>**

## 22 Summary

23 Represents a 32-bit unsigned integer.

## 24 Inherits From: System.ValueType

25

26 **Library:** BCL

27

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

29

## 30 Description

31 The `System.UInt32` data type represents integer values ranging from 0 to positive  
32 4,294,967,295 (hexadecimal 0xFFFFFFFF).

33

# 1 UInt32.MaxValue Field

```
2 [ILAsm]  
3 .field public static literal unsigned int32 MaxValue = 4294967295  
4 [C#]  
5 public const uint MaxValue = 4294967295
```

## 6 Summary

7 Contains the maximum value for the `System.UInt32` type.

## 8 Description

9 The value of this constant is 4,294,967,295 (hexadecimal `0xFFFFFFFF`).

10

# 1 UInt32.MinValue Field

```
2 [ILAsm]  
3 .field public static literal unsigned int32 MinValue = 0  
4 [C#]  
5 public const uint MinValue = 0
```

## 6 Summary

7 Contains the minimum value for the `System.UInt32` type.

## 8 Description

9 The value of this constant is 0.

10

# 1 UInt32.CompareTo(System.Object) Method

```
2 [ILAsm]  
3 .method public final hidebysig virtual int32 CompareTo(object value)  
4 [C#]  
5 public int CompareTo(object value)
```

## 6 Summary

7 Returns the sort order of the current instance compared to the specified `System.Object`.

## 8 Parameters

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

## 9 Return Value

11 The return value is a negative number, zero, or a positive number reflecting the sort  
12 order of the current instance as compared to *value*. For non-zero return values, the  
13 exact value returned by this method is unspecified. The following table defines the  
14 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.

## 15 Description

17 [Note: This method is implemented to support the `System.IComparable` interface.]  
18  
19

## 20 Exceptions

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

**System.ArgumentException**

*value* is not a *System.UInt32* and is not a null reference.

1

2

# 1 UInt32.CompareTo(System.UInt32) Method

```
2 [ILAsm]  
3 .method public final hidebysig virtual int32 CompareTo(unsigned int32  
4 value)  
5 [C#]  
6 public int CompareTo(uint value)
```

## 7 Summary

8 Returns the sort order of the current instance compared to the specified System.UInt32.

## 9 Parameters

Parameter	Description
<i>value</i>	The System.UInt32 to compare to the current instance.

## 10 Return Value

11 The return value is a negative number, zero, or a positive number reflecting the sort  
12 order of the current instance as compared to *value*. For non-zero return values, the  
13 exact value returned by this method is unspecified. The following table defines the  
14 return value:  
15

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> .

## 16 Description

17 [Note: This method is implemented to support the System.IComparable<UInt32>  
18 interface.]  
19  
20  
21  
22

# 1 UInt32.Equals(System.Object) Method

```
2 [ILAsm]  
3 .method public hidebysig virtual bool Equals(object obj)  
4 [C#]  
5 public override bool Equals(object obj)
```

## 6 Summary

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

## 9 Parameters

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

10

## 11 Return Value

12 `true` if *obj* represents the same type and value as the current instance. If *obj* is a null  
13 reference or is not an instance of `System.UInt32`, returns `false`.

## 14 Description

15 [Note: This method overrides `System.Object.Equals`.]  
16  
17

18

# 1 `UInt32.Equals(System.UInt32)` Method

```
2 [ILAsm]  
3 .method public hidebysig virtual bool Equals(unsigned int32 obj)  
4 [C#]  
5 public override bool Equals(uint obj)
```

## 6 **Summary**

7 Determines whether the current instance and the specified `System.UInt32` represent the  
8 same value.

## 9 **Parameters**

Parameter	Description
<i>obj</i>	The <code>System.UInt32</code> to compare to the current instance.

10

## 11 **Return Value**

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

## 13 **Description**

14 [Note: This method is implemented to support the `System.IEquatable<UInt32>`  
15 interface.]  
16  
17

18

# 1 UInt32.GetHashCode() Method

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

## 6 Summary

7 Generates a hash code for the current instance.

## 8 Return Value

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

## 10 Description

11 The algorithm used to generate the hash code is unspecified.

12

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

14

15

16

# 1 UInt32.Parse(System.String) Method

```
2 [ILAsm]  
3 .method public hidebysig static unsigned int32 Parse(string s)  
4 [C#]  
5 public static uint Parse(string s)
```

## 6 Summary

7 Returns the specified System.String converted to a System.UInt32 value.

## 8 Type Attributes:

- 9 • CLSCompliantAttribute(false)

## 10 Parameters

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

11

## 12 Return Value

13 The System.UInt32 value obtained from s.

## 14 Description

15 This version of System.UInt32.Parse is equivalent to System.UInt32.Parse(s,  
16 System.Globalization.NumberStyles.Integer, null).

17

18 The string s is parsed using the formatting information in a  
19 System.Globalization.NumberFormatInfo initialized for the current system culture.

20 [Note: For more information, see

21 System.Globalization.NumberFormatInfo.CurrentInfo.]

22

23

24

25 This method is not CLS-compliant. For a CLS-compliant alternative use  
26 System.Int64.Parse(System.String).

## 27 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.UInt32.MaxValue or less than System.UInt32.MinValue.

1

## 2 Example

3 This example demonstrates parsing a string to a System.UInt32.

4

5 [C#]

```
6 using System;
7 public class UInt32ParseClass {
8     public static void Main() {
9         string str = " 100 ";
10        Console.WriteLine("String: \"{0}\" <UInt32> {1}",str,UInt32.Parse(str));
11    }
12 }
```

13 The output is

14

15 String: " 100 " <UInt32> 100

16

# 1 `UInt32.Parse(System.String,` 2 `System.Globalization.NumberStyles)` Method

```
3 [ILAsm]  
4 .method public hidebysig static unsigned int32 Parse(string s, valuetype  
5 System.Globalization.NumberStyles style)
```

```
6 [C#]  
7 public static uint Parse(string s, NumberStyles style)
```

## 8 Summary

9 Returns the specified `System.String` converted to a `System.UInt32` value.

## 10 Type Attributes:

- 11 • `CLSCompliantAttribute(false)`

## 12 Parameters

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

## 14 Return Value

15 The `System.UInt32` value obtained from `s`.

## 16 Description

17 This version of `System.UInt32.Parse` is equivalent to `System.UInt32.Parse(s, style,`  
18 `null)`.

19  
20 The string is parsed using the formatting information in a  
21 `System.Globalization.NumberFormatInfo` initialized for the current system culture.  
22 [Note: For more information, see  
23 `System.Globalization.NumberFormatInfo.CurrentInfo`.]  
24  
25  
26

1 This method is not CLS-compliant. For a CLS-compliant alternative use  
2 `System.Int64.Parse(System.String, System.Globalization.NumberStyles)`.

3 **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.UInt32.MaxValue</code> or less than <code>System.UInt32.MinValue</code> .

4

5

# 1 UInt32.Parse(System.String, 2 System.IFormatProvider) Method

```
3 [ILAsm]  
4 .method public hidebysig static unsigned int32 Parse(string s, class  
5 System.IFormatProvider provider)  
  
6 [C#]  
7 public static uint Parse(string s, IFormatProvider provider)
```

## 8 Summary

9 Returns the specified System.String converted to a System.UInt32 value.

## 10 Type Attributes:

- 11 • CLSCompliantAttribute(false)

## 12 Parameters

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

13

## 14 Return Value

15 The System.UInt32 value obtained from *s*.

## 16 Description

17 This version of System.UInt32.Parse is equivalent to System.UInt32.Parse(*s*,  
18 System.Globalization.NumberStyles.Integer, *provider*).

19

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

24

25 This method is not CLS-compliant. For a CLS-compliant alternative use  
26 System.Int64.Parse (System.String, System.IFormatProvider).

## 27 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.UInt32.MaxValue</code> or less than <code>System.UInt32.MinValue</code> .

1

2

# 1 UInt32.Parse(System.String, 2 System.Globalization.NumberStyles, 3 System.IFormatProvider) Method

```
4 [ILAsm]  
5 .method public hidebysig static unsigned int32 Parse(string s, valuetype  
6 System.Globalization.NumberStyles style, class System.IFormatProvider  
7 provider)  
  
8 [C#]  
9 public static uint Parse(string s, NumberStyles style, IFormatProvider  
10 provider)
```

## 11 Summary

12 Returns the specified System.String converted to a System.UInt32 value.

## 13 Type Attributes:

- 14 • CLSCompliantAttribute(false)

## 15 Parameters

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

## 16 17 Return Value

18 The System.UInt32 value obtained from *s*.

## 19 Description

20 The string *s* is parsed using the culture-specific formatting information from the  
21 System.Globalization.NumberFormatInfo instance supplied by *provider*. If *provider* is

1 null or a `System.Globalization.NumberFormatInfo` cannot be obtained from *provider*,  
2 the formatting information for the current system culture is used.  
3  
4 This method is not CLS-compliant. For a CLS-compliant alternative use  
5 `System.Int64.Parse(System.String, System.Globalization.NumberStyles,`  
6 `System.IFormatProvider)`.

7 **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.UInt32.MaxValue</code> or less than <code>System.UInt32.MinValue</code> .

8

9

# 1 UInt32.ToString(System.IFormatProvider) 2 Method

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

## 8 Summary

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

## 10 Parameters

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

11

## 12 Return Value

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

## 16 Description

17 This version of `System.UInt32.ToString` is equivalent to  
18 `System.UInt32.ToString("G", provider)`.

19

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

22

# 1 UInt32.ToString(System.String, 2 System.IFormatProvider) Method

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

```
6 [C#]  
7 public string ToString(string format, IFormatProvider provider)
```

## 8 Summary

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

## 10 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.

## 12 Return Value

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

## 16 Description

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

19  
20 If *format* is a null reference the general format specifier "G" is used.

21  
22 [Note: For a detailed description of formatting, see the `System.IFormattable` interface.

23  
24 This method is implemented to support the `System.IFormattable` interface.

25  
26 ]

27  
28 The following table lists the characters that are valid for the `System.UInt32` 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.

1

## 2 Exceptions

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

3

4

# 1 UInt32.ToString() Method

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

## 6 Summary

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

## 8 Return Value

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

## 11 Description

12 This version of `System.UInt32.ToString` is equivalent to `System.UInt32.ToString`  
13 `(null, null)`.

14 [*Note:* This method overrides `System.Object.ToString`.]  
15  
16  
17

18

# 1 UInt32.ToString(System.String) Method

```
2 [ILAsm]  
3 .method public hidebysig instance string ToString(string format)  
4 [C#]  
5 public string ToString(string format)
```

## 6 Summary

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

## 8 Parameters

Parameter	Description
<i>format</i>	A <code>System.String</code> that specifies the format of the returned string. [ <i>Note:</i> For a list of valid values, see <code>System.UInt32.ToString(System.String, System.IFormatProvider)</code> .]

9

## 10 Return Value

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

## 13 Description

14 This method is equivalent to `System.UInt32.ToString(format, null)`.

15

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

## 17 Exceptions

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

18

## 19 Example

20 This example demonstrates converting a `System.UInt32` to a string.

21

22 [C#]

```
23 using System;  
24 public class UInt32ToStringExample {  
25     public static void Main() {
```

```
1     UInt32 i = 32;
2     Console.WriteLine(i);
3     String[] formats = {"c", "d", "e", "f", "g", "n", "p", "x" };
4     foreach(String str in formats)
5         Console.WriteLine("{0}: {1}", str, i.ToString(str));
6     }
7 }
```

8 The output is

```
9
10 32
11
12
13 c: $32.00
14
15
16 d: 32
17
18
19 e: 3.200000e+001
20
21
22 f: 32.00
23
24
25 g: 32
26
27
28 n: 32.00
29
30
31 p: 3,200.00 %
32
33
34 x: 20
35
36
```