

System.UInt32 Structure

```
[ILASM]
.class public sequential sealed serializable UInt32 extends
System.ValueType implements System.IComparable,
System.IFormattable

[C#]
public struct UInt32: IComparable, IFormattable
```

Assembly Info:

- *Name:* mscorlib
- *Public Key:* [00 00 00 00 00 00 00 00 04 00 00 00 00 00 00 00]
- *Version:* 1.0.x.x
- *Attributes:*
 - CLSCompliantAttribute(true)

Type Attributes:

- CLSCompliantAttribute(false)

Implements:

- **System.IComparable**
- **System.IFormattable**

Summary

22

23 Represents a 32-bit unsigned integer.

Inherits From: System.ValueType

25

26 **Library:** BCL

27

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

29

Description

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

33

1 UInt32.MaxValue Field

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

7 Summary

8 Contains the maximum value for the **System.UInt32** type.

9 Description

10 The value of this constant is 4,294,967,295 (hexadecimal
11 0xFFFFFFFF).

12

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) 2 Method

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

8 Summary

9 Returns the sort order of the current instance compared to the
10 specified **System.Object**.

11 Parameters

12
13

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

14
15
16

Return Value

17 A **System.Int32** containing a value that reflects the sort order of the
18 current instance as compared to *value*. The following table defines the
19 conditions under which the return value is a negative number, zero, or
20 a positive number.

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

21

22 Description

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

25 Exceptions

26
27

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

1
2
3

System.ArgumentException	<i>value</i> is not a System.UInt32 and is not a null reference.
---------------------------------	---

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
8 **System.Object** represent the same type and value.

9 **Parameters**

10
11

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

12
13
14

13 **Return Value**

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

18 **Description**

19 [Note: This method overrides **System.Object.Equals**.]
20

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

10 A **System.Int32** containing the hash code for the current instance.

11 Description

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

13

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

15

1 UInt32.Parse(System.String) Method

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

7 Summary

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

10 Type Attributes:

- 11 • CLSCompliantAttribute(false)

12 Parameters

13 Parameter	14 Description
s	A System.String containing the value to convert. The string is interpreted using the System.Globalization.NumberStyles.Integer style.

15 Return Value

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

17 Description

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

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

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

26 Exceptions

27 Exception	28 Condition
--------------	--------------

System.ArgumentNullException	s is a null reference.
System.FormatException	s is not in the correct style.
System.OverflowException	s represents a number greater than System.UInt32.MaxValue or less than System.UInt32.MinValue .

1
2
3

Example

4
5
6

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

```

7     using System;
8     public class UInt32ParseClass {
9         public static void Main() {
10            string str = " 100 ";
11            Console.WriteLine("String: \"{0}\" <UInt32>
12            {1}",str,UInt32.Parse(str));
13        }
14    }

```

15
16
17

The output is

```
String: " 100 " <UInt32> 100
```

18

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

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

9 **Summary**

10 Returns the specified **System.String** converted to a **System.UInt32**
11 value.

12 **Type Attributes:**

- 13 • CLSCompliantAttribute(false)

14 **Parameters**

15
16

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.

17
18
19

18 **Return Value**

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

21 **Description**

22 This version of **System.UInt32.Parse** is equivalent to
23 **System.UInt32.Parse(s, style, null)**.

24
25 The string is parsed using the formatting information in a
26 **System.Globalization.NumberFormatInfo** initialized for the current
27 system culture. [Note: For more information, see
28 **System.Globalization.NumberFormatInfo.CurrentInfo**.]
29

30 This method is not CLS-compliant. For a CLS-compliant alternative use

1 **System.Int64.Parse(System.String,**
2 **System.Globalization.NumberStyles).**

3 **Exceptions**

4
5

Exception	Condition
System.ArgumentNullException	s is a null reference.
System.FormatException	s is not in the correct style.
System.OverflowException	s represents a number greater than System.UInt32.MaxValue or less than System.UInt32.MinValue .

6
7
8

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

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

9 **Summary**

10 Returns the specified **System.String** converted to a **System.UInt32**
11 value.

12 **Type Attributes:**

- 13 • CLSCompliantAttribute(false)

14 **Parameters**

15
16

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> .

17
18
19

18 **Return Value**

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

21 **Description**

22 This version of **System.UInt32.Parse** is equivalent to
23 **System.UInt32.Parse(s,**
24 **System.Globalization.NumberStyles.Integer, provider).**

25
26 The string *s* is parsed using the culture-specific formatting information
27 from the **System.Globalization.NumberFormatInfo** instance
28 supplied by *provider*. If *provider* is **null** or a
29 **System.Globalization.NumberFormatInfo** cannot be obtained from
30 *provider*, the formatting information for the current system culture is
31 used.

1
2
3
4
5
6

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

Exceptions

Exception	Condition
System.ArgumentNullException	s is a null reference.
System.FormatException	s is not in the correct style.
System.OverflowException	s represents a number greater than System.UInt32.MaxValue or less than System.UInt32.MinValue .

7
8
9

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

```
4 [ILASM]  
5 .method public hidebysig static unsigned int32 Parse(string  
6 s, valuetype System.Globalization.NumberStyles style, class  
7 System.IFormatProvider provider)
```

```
8 [C#]  
9 public static uint Parse(string s, NumberStyles style,  
10 IFormatProvider provider)
```

11 Summary

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

14 Type Attributes:

- 15 • CLSCompliantAttribute(false)

16 Parameters

17
18

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> .

19
20
21

20 Return Value

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

23 Description

24 The string *s* is parsed using the culture-specific formatting information
25 from the **System.Globalization.NumberFormatInfo** instance
26 supplied by *provider*. If *provider* is **null** or a
27 **System.Globalization.NumberFormatInfo** cannot be obtained from

1 *provider*, the formatting information for the current system culture is
2 used.

3
4 This method is not CLS-compliant. For a CLS-compliant alternative use
5 **System.Int64.Parse(System.String,**
6 **System.Globalization.NumberStyles, System.IFormatProvider).**

7 **Exceptions**

8
9

Exception	Condition
System.ArgumentNullException	s is a null reference.
System.FormatException	s is not in the correct style.
System.OverflowException	s represents a number greater than System.UInt32.MaxValue or less than System.UInt32.MinValue .

10
11
12

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

```
3 [ILASM]  
4 .method public final hidebysig virtual string  
5 ToString(class 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
10 instance.

11 **Parameters**

12
13

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

14
15
16

17 **Return Value**

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

22 **Description**

23 This version of **System.UInt32.ToString** is equivalent to
24 **System.UInt32.ToString("G", provider)**.

25
26 If *provider* is **null** or a **System.Globalization.NumberFormatInfo**
27 cannot be obtained from *provider*, the formatting information for the
28 current system culture is used.

29

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

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

10 Summary

11 Returns a **System.String** representation of the value of the current
12 instance.

13 Parameters

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

16 Return Value

17 A **System.String** representation of the current instance formatted as
18 specified by *format*. The string takes into account the formatting
19 information in the **System.Globalization.NumberFormatInfo**
20 instance supplied by *provider*.
21
22

23 Description

24 If *provider* is **null** or a **System.Globalization.NumberFormatInfo**
25 cannot be obtained from *provider*, the formatting information for the
26 current system culture is used.
27

28 If *format* is a null reference the general format specifier "G" is used.
29

30 [Note: For a detailed description of formatting, see the
31 **System.IFormattable** interface.
32

33 This method is implemented to support the **System.IFormattable**
34 interface.] The following table lists the characters that are valid for the
35 **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**

3

4

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

5

6

7

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
8 instance.

9 Return Value

10

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

14 Description

15 This version of **System.UInt32.ToString** is equivalent to
16 **System.UInt32.ToString (null, null)**.

17

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

19

UInt32.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 System.String that specifies the format of the returned string. [Note: For a list of valid values, see System.UInt32.ToString(System.String, System.IFormatProvider) .]

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 method is equivalent to **System.UInt32.ToString** (*format*, **null**).

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

Exceptions

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

Example

This example demonstrates converting a **System.UInt32** to a string.

```
[C#]
```

```
1      using System;
2      public class UInt32ToStringExample {
3          public static void Main() {
4              UInt32 i = 32;
5              Console.WriteLine(i);
6              String[] formats = {"c", "d", "e", "f", "g", "n",
7 "p", "x" };
8              foreach(String str in formats)
9                  Console.WriteLine("{0}: {1}", str,
10 i.ToString(str));
11          }
12      }
```

13 The output is

14 32

15

16

17

18 c: \$32.00

19

20

21 d: 32

22

23

24 e: 3.200000e+001

25

26

27 f: 32.00

28

29

30 g: 32

31

32

1 n: 32.00
2
3
4 p: 3,200.00 %
5
6
7 x: 20
8
9