

System.Double Structure

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

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

Assembly Info:

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

Implements:

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

Summary

Represents a 64-bit double-precision floating-point number.

Inherits From: System.ValueType

Library: ExtendedNumerics

Thread Safety: This type is safe for multithreaded operations.

Description

System.Double is a 64-bit double precision floating-point type that represents values ranging from approximately 5.0E-324 to 1.7E+308 and from approximately -5.0E-324 to -1.7E+308 with a precision of 15-16 decimal digits. The **System.Double** type conforms to standard IEC 60559:1989, Binary Floating-point Arithmetic for Microprocessor Systems.

A **System.Double** can represent the following values:

- The finite set of non-zero values of the form $s * m * 2^e$, where s is 1 or -1, and $0 < m < 2^{53}$ and $-1075 \leq e \leq 970$.

- 1 • Positive infinity and negative infinity. Infinities are produced by
2 operations that produce results with a magnitude greater than
3 that which can be represented by a **System.Double**, such as
4 dividing a non-zero number by zero. For example, using
5 **System.Double** operands, $1.0 / 0.0$ yields positive infinity,
6 and $-1.0 / 0.0$ yields negative infinity. Operations include
7 passing parameters and returning values.
- 8 • The *Not-a-Number* value (NaN). NaN values are produced by
9 invalid floating-point operations, such as dividing zero by zero.

10 When performing binary operations, if one of the operands is a
11 **System.Double**, then the other operand is required to be an integral
12 type or a floating-point type (**System.Double** or **System.Single**).
13 Prior to performing the operation, if the other operand is not a
14 **System.Double**, it is converted to **System.Double**, and the
15 operation is performed using at least **System.Double** range and
16 precision. If the operation produces a numeric result, the type of the
17 result is **System.Double**.

18 The floating-point operators, including the assignment operators, do
19 not throw exceptions. Instead, in exceptional situations, the result of a
20 floating-point operation is zero, infinity, or NaN, as described below:
21

- 22 • If the result of a floating-point operation is too small for the
23 destination format, the result of the operation is zero.
- 24 • If the magnitude of the result of a floating-point operation is
25 too large for the destination format, the result of the operation
26 is positive infinity or negative infinity, as appropriate for the
27 sign of the result.
- 28 • If a floating-point operation is invalid, the result of the
29 operation is NaN.
- 30 • If one or both operands of a floating-point operation are NaN,
31 the result of the operation is NaN.

32 Conforming implementations of the CLI are permitted to perform
33 floating-point operations using a precision that is higher than that
34 required by the **System.Double** type. For example, hardware
35 architectures that support an "extended" or "long double" floating-
36 point type with greater range and precision than the **System.Double**
37 type could implicitly perform all floating-point operations using this
38 higher precision type. Expressions evaluated using a higher precision
39 may cause a finite result to be produced instead of an infinity.

40

1 Double.Epsilon Field

```
2 [ILASM]  
3 .field public static literal float64 Epsilon =  
4 4.9406564584124654e-324  
5 [C#]  
6 public const double Epsilon = 4.9406564584124654e-324
```

7 Summary

8 Represents the smallest positive **System.Double** value greater than
9 zero.

10 Description

11 The value of this constant is 4.9406564584124654E-324.

12

1 Double.MaxValue Field

```
2 [ILASM]  
3 .field public static literal float64 MaxValue =  
4 1.7976931348623157e+308  
  
5 [C#]  
6 public const double MaxValue = 1.7976931348623157e+308
```

7 Summary

8 Contains the maximum positive value for the **System.Double** type.

9 Description

10 The value of this constant is 1.7976931348623157E+308.

11

1 Double.MinValue Field

```
2 [ILASM]  
3 .field public static literal float64 MinValue = -  
4 1.7976931348623157e+308  
5 [C#]  
6 public const double MinValue = -1.7976931348623157e+308
```

7 Summary

8 Contains the minimum (most negative) value for the **System.Double**
9 type.

10 Description

11 The value of this constant is -1.7976931348623157E+308.

12

1 Double.NaN Field

```
2 [ILASM]  
3 .field public static literal float64 NaN = (double)0.0 /  
4 (double)0.0  
5 [C#]  
6 public const double NaN = (double)0.0 / (double)0.0
```

7 Summary

8 Represents an undefined result of operations involving
9 **System.Double**.

10 Description

11 Not-a-Number (NaN) values are returned when the result of a
12 **System.Double** operation is undefined.

13
14 A NaN value is not equal to any other value, including another NaN
15 value.

16
17 The value of this field is obtained by dividing **System.Double** zero by
18 zero.

19
20 [*Note:* **System.Double.NaN** represents one of many possible NaN
21 values. To test whether a **System.Double** value is a NaN, use the
22 **System.Double.IsNaN** method.]

23

1 Double.NegativeInfinity Field

```
2 [ILASM]  
3 .field public static literal float64 NegativeInfinity =  
4 (double)-1.0 / (double)(0.0)
```

```
5 [C#]  
6 public const double NegativeInfinity = (double)-1.0 /  
7 (double)(0.0)
```

8 Summary

9 Represents a negative infinity of type **System.Double**.

10 Description

11 The value of this constant is obtained by dividing a negative
12 **System.Double** by zero.

13
14 [Note: To test whether a **System.Double** value is a negative infinity
15 value, use the **System.Double.IsNegativeInfinity** method.]

16

1 Double.PositiveInfinity Field

```
2 [ILASM]  
3 .field public static literal float64 PositiveInfinity =  
4 (double)1.0 / (double)(0.0)
```

```
5 [C#]  
6 public const double PositiveInfinity = (double)1.0 /  
7 (double)(0.0)
```

8 Summary

9 Represents a positive infinity of type **System.Double**.

10 Description

11 The value of this constant is obtained by dividing a positive
12 **System.Double** by zero.

13
14 [*Note:* To test whether a **System.Double** value is a positive infinity
15 value, use the **System.Double.IsPositiveInfinity** method.]

16

1 Double.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 returned value is a negative number, zero,
20 or a positive number.

Value	Description
Any negative number	Current instance < <i>value</i> . Current instance is a NaN and <i>value</i> is not a NaN and is not a null reference.
Zero	Current instance == <i>value</i> . Current instance and <i>value</i> are both NaN, positive infinity, or negative infinity.
Any positive number	Current instance > <i>value</i> . <i>value</i> is a null reference. Current instance is not a NaN and <i>value</i> is a NaN.

21

1 **Description**

2 [Note: This method is implemented to support the
3 **System.IComparable** interface. Note that, although a NaN is not
4 considered to be equal to another NaN (even itself), the
5 **System.IComparable** interface requires that A.CompareTo (A)
6 return zero.]

7 **Exceptions**

8
9

Exception	Condition
System.ArgumentException	<i>value</i> is not a null reference and is not of type System.Double .

10
11
12

1 Double.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

Return Value

15 **true** if *obj* is a **System.Double** with the same value as the current
16 instance, otherwise **false**. If *obj* is a null reference or is not an
17 instance of **System.Double**, returns **false**. If either *obj* or the current
18 instance is a NaN and the other is not, returns **false**. If *obj* and the
19 current instance are both NaN, positive infinity, or negative infinity,
20 returns **true**.

21 Description

22 [Note: This method overrides **System.Object.Equals**.]
23

1 Double.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 this 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 Double.IsInfinity(System.Double) Method

```
2 [ILASM]  
3 .method public hidebysig static bool IsInfinity(float64 d)  
4 [C#]  
5 public static bool IsInfinity(double d)
```

6 Summary

7 Determines whether the specified **System.Double** represents an
8 infinity, which can be either positive or negative.

9 Parameters

10
11

Parameter	Description
<i>d</i>	The System.Double to be checked.

12
13
14

Return Value

15 **true** if *d* represents a positive or negative infinity value; otherwise
16 **false**.

17 Description

18 [Note: Floating-point operations return positive or negative infinity
19 values to signal an overflow condition.]

20

1 Double.IsNaN(System.Double) Method

```
2 [ILASM]  
3 .method public hidebysig static bool IsNaN(float64 d)  
4 [C#]  
5 public static bool IsNaN(double d)
```

6 Summary

7 Determines whether the value of the specified **System.Double** is
8 undefined (Not-a-Number).

9 Parameters

10
11

Parameter	Description
<i>d</i>	The System.Double to be checked.

12
13
14

Return Value

15 **true** if *d* represents a NaN value; otherwise **false**.

16 Description

17 [Note: Floating-point operations return NaN values to signal that the
18 result of the operation is undefined. For example, dividing (Double)
19 0.0 by 0.0 results in a NaN value.]

20

1 Double.IsNegativeInfinity(System.Double 2) Method

```
3 [ILASM]  
4 .method public hidebysig static bool  
5 IsNegativeInfinity(float64 d)  
  
6 [C#]  
7 public static bool IsNegativeInfinity(double d)
```

8 Summary

9 Determines whether the specified **System.Double** represents a
10 negative infinity value.

11 Parameters

12
13

Parameter	Description
<i>d</i>	The System.Double to be checked.

14
15
16

Return Value

17 **true** if *d* represents a negative infinity value; otherwise **false**.

18 Description

19 [Note: Floating-point operations return negative infinity values to
20 signal an overflow condition.]

21

1 Double.IsPositiveInfinity(System.Double)

2 Method

```
3 [ILASM]  
4 .method public hidebysig static bool  
5 IsPositiveInfinity(float64 d)  
  
6 [C#]  
7 public static bool IsPositiveInfinity(double d)
```

8 Summary

9 Determines whether the specified **System.Double** represents a
10 positive infinity value.

11 Parameters

12
13

Parameter	Description
<i>d</i>	The System.Double to be checked.

14
15
16

15 Return Value

17 **true** if *d* represents a positive infinity value; otherwise **false**.

18 Description

19 [Note: Floating-point operations return positive infinity values to signal
20 an overflow condition.]

21

1 Double.Parse(System.String) Method

```
2 [ILASM]  
3 .method public hidebysig static float64 Parse(string s)  
4  
5 [C#]  
6 public static double Parse(string s)
```

6 Summary

7 Returns the specified **System.String** converted to a **System.Double**
8 value.

9 Parameters

10
11

Parameter	Description
s	A System.String containing the value to convert. The string is interpreted using the System.Globalization.NumberStyles.Float and/or System.Globalization.NumberStyles.AllowThousands style.

12
13
14

13 Return Value

15 The **System.Double** value obtained from s. If the parsed value is less
16 than **System.Double.MinValue**, this method returns
17 **System.Double.NegativeInfinity**. If the parsed value is greater than
18 **System.Double.MaxValue**, this method returns
19 **System.Double.PositiveInfinity**. If s equals
20 **System.Globalization.NumberFormatInfo.NaNSymbol**, this
21 method returns **System.Double.NaN**.

22 Description

23 This version of **System.Double.Parse** is equivalent to
24 **System.Double.Parse** (s,
25 **System.Globalization.NumberStyles.Float** |
26 **System.Globalization.NumberStyles.AllowThousands**, null).

27
28
29
30
31

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**.]

32 Exceptions

33
34

Exception	Condition
System.ArgumentNullException	s is a null reference.

1
2
3

System.FormatException	s is not in the correct style.
-------------------------------	--------------------------------

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

```
4 [ILASM]  
5 .method public hidebysig static float64 Parse(string s,  
6 valuetype System.Globalization.NumberStyles style)  
  
7 [C#]  
8 public static double Parse(string s, NumberStyles style)
```

9 Summary

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

12 Parameters

13
14

Parameter	Description
s	A System.String containing the value to convert. The string is interpreted using the style specified by <i>style</i> .
style	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.Float and System.Globalization.NumberStyles.AllowThousands styles.

15 16 Return Value

17
18 The **System.Double** value obtained from *s*. If the parsed value is less
19 than **System.Double.MinValue**, this method returns
20 **System.Double.NegativeInfinity**. If the parsed value is greater than
21 **System.Double.MaxValue**, this method returns
22 **System.Double.PositiveInfinity**. If *s* equals
23 **System.Globalization.NumberFormatInfo.NaNSymbol**, this
24 method returns **System.Double.NaN**.

25 Description

26 This version of **System.Double.Parse** is equivalent to
27 **System.Double.Parse** (*s*, *style*, **null**).
28
29 The string *s* is parsed using the formatting information in a
30 **System.Globalization.NumberFormatInfo** initialized for the current

1 system culture. [*Note:* For more information, see
2 **System.Globalization.NumberFormatInfo.CurrentInfo.**]

3 **Exceptions**

4
5

Exception	Condition
System.ArgumentNullException	s is a null reference.
System.FormatException	s is not in the correct style.

6
7
8

1 Double.Parse(System.String, 2 System.IFormatProvider) Method

```
3 [ILASM]  
4 .method public hidebysig static float64 Parse(string s,  
5 class System.IFormatProvider provider)  
  
6 [C#]  
7 public static double Parse(string s, IFormatProvider  
8 provider)
```

9 Summary

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

12 Parameters

13
14

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

15
16
17

Return Value

18 The **System.Double** value obtained from *s*. If the parsed value is less
19 than **System.Double.MinValue**, this method returns
20 **System.Double.NegativeInfinity**. If the parsed value is greater than
21 **System.Double.MaxValue**, this method returns
22 **System.Double.PositiveInfinity**. If *s* equals
23 **System.Globalization.NumberFormatInfo.NaNSymbol**, this
24 method returns **System.Double.NaN**.

25 Description

26 This version of **System.Double.Parse** is equivalent to
27 **System.Double.Parse**(*s*,
28 **System.Globalization.NumberStyles.Float** |
29 **System.Globalization.NumberStyles.AllowThousands**, *provider*).

30
31 The string *s* is parsed using the culture-specific formatting information
32 from the **System.Globalization.NumberFormatInfo** instance
33 supplied by *provider*. If *provider* is **null** or a

1 **System.Globalization.NumberFormatInfo** cannot be obtained from
2 *provider*, the formatting information for the current system culture is
3 used.

4 **Exceptions**

5
6

Exception	Condition
System.ArgumentNullException	s is a null reference.
System.FormatException	s is not in the correct style.

7
8
9

Double.Parse(System.String, System.Globalization.NumberStyles, System.IFormatProvider) Method

```
[ILASM]  
.method public hidebysig static float64 Parse(string s,  
valuetype System.Globalization.NumberStyles style, class  
System.IFormatProvider provider)
```

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

Summary

Returns the specified **System.String** converted to a **System.Double** value.

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.Float and System.Globalization.NumberStyles.AllowThousands styles.
<i>provider</i>	A System.IFormatProvider that supplies a System.Globalization.NumberFormatInfo containing culture-specific formatting information about <i>s</i> .

Return Value

The **System.Double** value obtained from *s*. If the parsed value is less than **System.Double.MinValue**, this method returns **System.Double.NegativeInfinity**. If the parsed value is greater than **System.Double.MaxValue**, this method returns **System.Double.PositiveInfinity**. If *s* equals **System.Globalization.NumberFormatInfo.NaNSymbol**, this method returns **System.Double.NaN**.

Description

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

7 **Exceptions**
8
9

Exception	Condition
System.ArgumentNullException	<i>s</i> is a null reference
System.FormatException	<i>s</i> is not in the correct style.

10
11
12

1 Double.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

Return Value

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

22 Description

23 This version of **System.Double.ToString** is equivalent to
24 **System.Double.ToString (null, 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
30 [Note: The general format specifier formats the number in either fixed-
31 point or exponential notation form. For a detailed description of the
32 general format, see the **System.IFormattable** interface.]

33

1 Double.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, optionally followed by a non-negative integer that specifies the precision of the number in the returned System.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 information in
19 the **System.Globalization.NumberFormatInfo** instance supplied by
20 *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 The following table lists the format characters that are valid for the
31 **System.Double** type.

Format Characters	Description
-------------------	-------------

"C", "c"	Currency format.
"E", "e"	Exponential notation format.
"F", "f"	Fixed-point format.
"G", "g"	General format.
"N", "n"	Number format.
"P", "p"	Percent format.
"R", "r"	Round-trip format.

1

2

3

4

5

6

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

This method is implemented to support the **System.IFormattable** interface.]

7

Exceptions

8

9

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

10

11

12

1 Double.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.Double.ToString** is equivalent to
16 **System.Double.ToString (null, null)**.

17

18 [*Note:* The general format specifier formats the number in either fixed-
19 point or exponential notation form. For a detailed description of the
20 general format, see the **System.IFormattable** interface.

21

22 This method overrides **System.Object.ToString.**]

23

1 Double.ToString(System.String) Method

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

7 Summary

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

10 Parameters

11
12

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.Double.ToString(System.String, System.IFormatProvider) .]

13
14
15

14 Return Value

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

19 Description

20 This version of **System.Double.ToString** is equivalent to
21 **System.Double.ToString** (*format*, **null**).

22
23

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

24 Exceptions

25
26

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

27
28
29

28 Example

30 The following example shows the effects of various formats on the
31 string returned by **System.Double.ToString**.

```
1
2      [C#]
3      using System;
4      class test {
5          public static void Main() {
6              double d = 1234.56789;
7              Console.WriteLine(d);
8              string[] fmts = {"C", "E", "e5", "F", "G", "N", "P", "R"};
9              for (int i=0;i<fmts.Length;i++)
10                 Console.WriteLine("{0}: {1}",
11                    fmts[i],d.ToString(fmts[i]));
12            }
13        }
14
```

15 The output is

16 1234.56789

17
18
19 C: \$1,234.57

20
21
22 E: 1.234568E+003

23
24
25 e5: 1.23457e+003

26
27
28 F: 1234.57

29
30
31 G: 1234.56789

1
2
3
4
5
6
7
8
9
10
11

N: 1,234.57

P: 123,456.79 %

R: 1234.56789