

1 System.DateTime Structure

```
2 [ILAsm]
3 .class public sealed serializable DateTime extends System.ValueType
4 implements System.IComparable, System.IFormattable,
5 System.IComparable`1<valuetype System.DateTime>,
6 System.IEquatable`1<valuetype System.DateTime>
7
8 [C#]
9 public struct DateTime: IComparable, IFormattable, IComparable<DateTime>,
10 IEquatable<DateTime>
```

10 Assembly Info:

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

16 Implements:

- 17 • **System.IComparable**
- 18 • **System.IFormattable**
- 19 • **System.IComparable<System.DateTime>**
- 20 • **System.IEquatable<System.DateTime>**

21 Summary

22 Represents an instant in time, expressed as a date and time of day.

23 Inherits From: System.ValueType

24 **Library:** BCL

25 **Thread Safety:** All public static members of this type are safe for multithreaded operations.
26 No instance members are guaranteed to be thread safe.

30 Description

31 The *System.DateTime* value type represents dates and times with values ranging from
32 00:00:00, 1/1/0001 (Common Era) to 23:59:59 PM, 12/31/9999.

33
34 [*Note:* Time values are measured in 100-nanosecond units, *ticks*, and a particular date
35 is the number of ticks since 12:00 Midnight, January 1, 1 in the Gregorian calendar. For
36 example, a ticks value of 312413760000000000L represents the date, Friday, January
37 01, 0100 12:00:00 AM.

38
39 Time values can be added to, or subtracted from, an instance of *System.DateTime*. Time

1 values can be negative or positive, and expressed in units such as ticks, seconds, or
2 instances of `System.TimeSpan`. Methods and properties in this value type take into
3 account details such as leap years and the number of days in a month.

4

5 12:00:00 AM is Midnight.

6

7]

8

DateTime(System.Int32, System.Int32, System.Int32, System.Int32, System.Int32, System.Int32) Constructor

```
[ILAsm]
public rtspecialname specialname instance void .ctor(int32 year, int32
month, int32 day, int32 hour, int32 minute, int32 second, int32
millisecond)

[C#]
public DateTime(int year, int month, int day, int hour, int minute, int
second, int millisecond)
```

Summary

Constructs and initializes a new instance of the `System.DateTime` structure with a specified year, month, day, hour, minute, second, and millisecond.

Parameters

Parameter	Description
<i>year</i>	A <code>System.Int32</code> containing the year (1 through 9999).
<i>month</i>	A <code>System.Int32</code> containing the month (1 through 12).
<i>day</i>	A <code>System.Int32</code> containing the day (1 through the number of days in <i>month</i>).
<i>hour</i>	A <code>System.Int32</code> containing the hours (0 through 23).
<i>minute</i>	A <code>System.Int32</code> containing the minutes (0 through 59).
<i>second</i>	A <code>System.Int32</code> containing the seconds (0 through 59).
<i>millisecond</i>	A <code>System.Int32</code> containing the milliseconds.

Exceptions

Exception	Condition
System.ArgumentOutOfRangeException	<i>year</i> is less than 1 or greater than 9999 -or- <i>month</i> is less than 1 or greater than 12

	<p>-or-</p> <p><i>day</i> is less than 1 or greater than the number of days in <i>month</i></p> <p>-or-</p> <p><i>hour</i> is less than 0 or greater than 23</p> <p>-or-</p> <p><i>minute</i> is less than 0 or greater than 59</p> <p>-or-</p> <p><i>second</i> is less than 0 or greater than 59</p> <p>-or-</p> <p><i>millisecond</i> is less than 0 or greater than 999</p>
<p>System.ArgumentException</p>	<p>The specified parameters evaluate to a date less than <code>System.DateTime.MinValue</code> or greater than <code>System.DateTime.MaxValue</code>.</p>

1

2

DateTime(System.Int32, System.Int32, System.Int32, System.Int32, System.Int32) Constructor

```
[ILAsm]
public rtspecialname specialname instance void .ctor(int32 year, int32
month, int32 day, int32 hour, int32 minute, int32 second)

[C#]
public DateTime(int year, int month, int day, int hour, int minute, int
second)
```

Summary

Constructs and initializes a new instance of the `System.DateTime` structure with a specified year, month, day, hour, minute, and second.

Parameters

Parameter	Description
<i>year</i>	A <code>System.Int32</code> containing the year (1 through 9999).
<i>month</i>	A <code>System.Int32</code> containing the month (1 through 12).
<i>day</i>	A <code>System.Int32</code> containing the day (1 through the number of days in <i>month</i>).
<i>hour</i>	A <code>System.Int32</code> containing the hours (0 through 23).
<i>minute</i>	A <code>System.Int32</code> containing the minutes (0 through 59).
<i>second</i>	A <code>System.Int32</code> containing the seconds (0 through 59).

14

Exceptions

Exception	Condition
<code>System.ArgumentOutOfRangeException</code>	<i>year</i> is less than 1 or greater than 9999 -or- <i>month</i> is less than 1 or greater than 12 -or- <i>day</i> is less than 1 or greater than the number

of days in *month*

-or-

hour is less than 0 or greater than 23

-or-

minute is less than 0 or greater than 59

-or-

second is less than 0 or greater than 59

1

2

1 DateTime(System.Int64) Constructor

```
2 [ILAsm]  
3 public rtspecialname specialname instance void .ctor(int64 ticks)  
4 [C#]  
5 public DateTime(long ticks)
```

6 Summary

7 Constructs and initializes a new instance of the `System.DateTime` structure with the
8 date and time expressed in 100-nanosecond units.

9 Parameters

Parameter	Description
<i>ticks</i>	A <code>System.Int64</code> containing the date and time expressed in 100-nanosecond units.

10

11 Exceptions

Exception	Condition
System.ArgumentOutOfRangeException	The date and time represented by <i>ticks</i> is less than <code>System.DateTime.MinValue</code> or greater than <code>System.DateTime.MaxValue</code> .

12

13

1 DateTime(System.Int32, System.Int32, 2 System.Int32) Constructor

```
3 [ILAsm]  
4 public rtspecialname specialname instance void .ctor(int32 year, int32  
5 month, int32 day)  
  
6 [C#]  
7 public DateTime(int year, int month, int day)
```

8 Summary

9 Constructs and initializes a new instance of the `System.DateTime` structure with a
10 specified year, month, and day.

11 Parameters

Parameter	Description
<i>year</i>	A <code>System.Int32</code> containing the year (1 through 9999).
<i>month</i>	A <code>System.Int32</code> containing the month (1 through 12).
<i>day</i>	A <code>System.Int32</code> containing the day (1 through the number of days in <i>month</i>).

12 13 Description

14 The time of day for the resulting `System.DateTime` is midnight (00:00:00).

15 Exceptions

Exception	Condition
System.ArgumentOutOfRangeException	<i>year</i> is less than 1 or greater than 9999 -or- <i>month</i> is less than 1 or greater than 12 -or- <i>day</i> is less than 1 or greater than the number of days in <i>month</i>

16

1 DateTime.MaxValue Field

```
2 [ILAsm]  
3 .field public static initOnly valuetype System.DateTime MaxValue  
4 [C#]  
5 public static readonly DateTime MaxValue
```

6 Summary

7 A constant representing the largest possible value of System.DateTime.

8 Description

9 This field is read-only.

10
11 The value of this field is equivalent to 23:59:59.9999999, 12/31/9999, exactly one 100-
12 nanosecond tick before 00:00:00, 01/01/10000.

13

1 DateTime.MinValue Field

```
2 [ILAsm]  
3 .field public static initOnly valuetype System.DateTime MinValue  
4 [C#]  
5 public static readonly DateTime MinValue
```

6 Summary

7 A constant representing the smallest possible value of `System.DateTime`.

8 Description

9 This field is read-only.

10
11 The value of this field is equivalent to 00:00:00.0000000, 1/1/0001.

12

1 DateTime.Add(System.TimeSpan) Method

```
2 [ILAsm]  
3 .method public hidebysig instance valuetype System.DateTime Add(valuetype  
4 System.TimeSpan value)  
  
5 [C#]  
6 public DateTime Add(TimeSpan value)
```

7 Summary

8 Adds the value of a specified `System.TimeSpan` instance to the current instance.

9 Parameters

Parameter	Description
<i>value</i>	A <code>System.TimeSpan</code> instance.

10

11 Return Value

12 A `System.DateTime` instance set to the sum of the date and time of the current instance
13 and the time interval represented by *value*.

14 Description

15 A specified `System.TimeSpan` is added to the current instance of `System.DateTime`, and
16 the result is returned as a new `System.DateTime`.

17 Exceptions

Exception	Condition
System.ArgumentOutOfRangeException	The resulting <code>System.DateTime</code> is less than <code>System.DateTime.MinValue</code> or greater than <code>System.DateTime.MaxValue</code> .

18

19

1 **The following member must be implemented if the ExtendedNumerics library is present in**
2 **the implementation.**

3 DateTime.AddDays(System.Double) Method

```
4 [ILAsm]  
5 .method public hidebysig instance valuetype System.DateTime  
6 AddDays(float64 value)  
  
7 [C#]  
8 public DateTime AddDays(double value)
```

9 Summary

10 Adds a specified number of days to the value of the current instance.

11 Parameters

Parameter	Description
<i>value</i>	A <code>System.Double</code> containing the number of whole and fractional days. For example, 4.5 is equivalent to 4 days, 12 hours, 0 minutes, 0 seconds, 0 milliseconds, and 0 ticks. <i>value</i> can be negative or positive.

13 Return Value

14 A `System.DateTime` instance set to the sum of the date and time represented by the
15 current instance and the number of days represented by *value*.

16 Description

17 [Note: *value* is rounded to the nearest tick.]
18
19

20 Exceptions

Exception	Condition
System.ArgumentOutOfRangeException	The resulting <code>System.DateTime</code> is less than <code>System.DateTime.MinValue</code> or greater than <code>System.DateTime.MaxValue</code> .

The following member must be implemented if the ExtendedNumerics library is present in the implementation.

DateTime.AddHours(System.Double) Method

```
[ILAsm]  
.method public hidebysig instance valuetype System.DateTime  
AddHours(float64 value)  
  
[C#]  
public DateTime AddHours(double value)
```

Summary

Adds a specified number of hours to the value of the current instance.

Parameters

Parameter	Description
<i>value</i>	A <code>System.Double</code> containing the number of whole and fractional hours. For example, 4.5 is equivalent to 4 hours, 30 minutes, 0 seconds, 0 milliseconds, and 0 ticks. <i>value</i> can be negative or positive.

Return Value

A `System.DateTime` instance set to the sum of the date and time represented by the current instance and the number of hours represented by *value*.

Description

[Note: *value* is rounded to the nearest tick.]

Exceptions

Exception	Condition
System.ArgumentOutOfRangeException	The resulting <code>System.DateTime</code> is less than <code>System.DateTime.MinValue</code> or greater than <code>System.DateTime.MaxValue</code> .

1 **The following member must be implemented if the ExtendedNumerics library is present in**
2 **the implementation.**

3 DateTime.AddMilliseconds(System.Double) 4 Method

```
5 [ILAsm]  
6 .method public hidebysig instance valuetype System.DateTime  
7 AddMilliseconds(float64 value)  
8 [C#]  
9 public DateTime AddMilliseconds(double value)
```

10 Summary

11 Adds a specified number of milliseconds to the value of the current instance.

12 Parameters

Parameter	Description
<i>value</i>	A <code>System.Double</code> containing the number of whole and fractional milliseconds. For example, 4.5 is equivalent to 4 milliseconds and 5,000 ticks. <i>value</i> can be negative or positive.

14 Return Value

15 A `System.DateTime` instance set to the sum of the date and time represented by the
16 current instance and the number of milliseconds represented by *value*.

17 Description

18 [*Note: value* is rounded to the nearest tick.]
19
20

21 Exceptions

Exception	Condition
System.ArgumentOutOfRangeException	The resulting <code>System.DateTime</code> is less than <code>System.DateTime.MinValue</code> or greater than <code>System.DateTime.MaxValue</code> .

The following member must be implemented if the ExtendedNumerics library is present in the implementation.

DateTime.AddMinutes(System.Double) Method

```
[ILAsm]  
.method public hidebysig instance valuetype System.DateTime  
AddMinutes(float64 value)  
  
[C#]  
public DateTime AddMinutes(double value)
```

Summary

Adds a specified number of minutes to the value of the current instance.

Parameters

Parameter	Description
<i>value</i>	A <code>System.Double</code> containing the number of whole and fractional minutes. For example, 4.5 is equivalent to 4 minutes, 30 seconds, 0 milliseconds, and 0 ticks. <i>value</i> can be negative or positive.

Return Value

A `System.DateTime` instance set to the sum of the date and time represented by the current instance and the number of minutes represented by *value*.

Description

[*Note: value* is rounded to the nearest tick.]

Exceptions

Exception	Condition
System.ArgumentOutOfRangeException	The resulting <code>System.DateTime</code> is less than <code>System.DateTime.MinValue</code> or greater than <code>System.DateTime.MaxValue</code> .

1 **DateTime.AddMonths(System.Int32) Method**

```
2 [ILAsm]
3 .method public hidebysig instance valuetype System.DateTime
4 AddMonths(int32 months)
5
6 [C#]
public DateTime AddMonths(int months)
```

7 **Summary**

8 Adds a specified number of months to the value of the current instance.

9 **Parameters**

Parameter	Description
<i>months</i>	A System.Int32 containing the number of months. <i>months</i> can be positive or negative, and can be greater than the number of months in a year.

10
11 **Return Value**

12 A System.DateTime instance set to the sum of the date and time represented by the
13 current instance and *months*.

14 **Description**

15 This method does not change the value of the current DateTime instance. Instead, a
16 new DateTime instance is returned whose value is the result of this operation.

17 **Exceptions**

Exception	Condition
System.ArgumentOutOfRangeException	The resulting System.DateTime is less than System.DateTime.MinValue or greater than System.DateTime.MaxValue. -or- The <i>months</i> parameter is less than -120,000 or greater than 120,000

18
19

1 **The following member must be implemented if the ExtendedNumerics library is present in**
2 **the implementation.**

3 DateTime.AddSeconds(System.Double) 4 Method

```
5 [ILAsm]  
6 .method public hidebysig instance valuetype System.DateTime  
7 AddSeconds(float64 value)  
8 [C#]  
9 public DateTime AddSeconds(double value)
```

10 Summary

11 Adds a specified number of seconds to the value of the current instance.

12 Parameters

Parameter	Description
<i>value</i>	A <code>System.Double</code> containing the number of whole and fractional seconds. For example, 4.5 is equivalent to 4 seconds, 500 milliseconds, and 0 ticks. <i>value</i> can be positive or negative.

14 Return Value

15 A `System.DateTime` instance set to the sum of the date and time represented by the
16 current instance and the number of seconds represented by *value*.

17 Description

18 [*Note: value* is rounded to the nearest tick.]
19
20

21 Exceptions

Exception	Condition
System.ArgumentException	The resulting <code>System.DateTime</code> is less than <code>System.DateTime.MinValue</code> or greater than <code>System.DateTime.MaxValue</code> .

1 DateTime.AddTicks(System.Int64) Method

```
2 [ILAsm]  
3 .method public hidebysig instance valuetype System.DateTime AddTicks(int64  
4 value)  
5 [C#]  
6 public DateTime AddTicks(long value)
```

7 Summary

8 Adds a specified number of ticks to the value of the current instance.

9 Parameters

Parameter	Description
<i>value</i>	A <code>System.Int64</code> containing the number of 100-nanosecond ticks. <i>value</i> can be positive or negative.

10

11 Return Value

12 A `System.DateTime` instance set to the sum of the date and time represented by the
13 current instance and the time represented by *value*.

14 Exceptions

Exception	Condition
System.ArgumentOutOfRangeException	The resulting <code>System.DateTime</code> is less than <code>System.DateTime.MinValue</code> or greater than <code>System.DateTime.MaxValue</code> .

15

16

1 DateTime.AddYears(System.Int32) Method

```
2 [ILAsm]  
3 .method public hidebysig instance valuetype System.DateTime AddYears(int32  
4 value)  
5 [C#]  
6 public DateTime AddYears(int value)
```

7 Summary

8 Adds a specified number of years to the value of the current instance.

9 Parameters

Parameter	Description
<i>value</i>	A <code>System.Int32</code> containing the number of years. <i>value</i> can be positive or negative.

10

11 Return Value

12 A `System.DateTime` instance set to the sum of the date and time represented by the
13 current instance and the number of years represented by *value*.

14 Exceptions

Exception	Condition
System.ArgumentOutOfRangeException	The resulting <code>System.DateTime</code> is less than <code>System.DateTime.MinValue</code> or greater than <code>System.DateTime.MaxValue</code> .

15

16

1 DateTime.Compare(System.DateTime, 2 System.DateTime) Method

```
3 [ILAsm]  
4 .method public hidebysig static int32 Compare(valuetype System.DateTime  
5 t1, valuetype System.DateTime t2)  
  
6 [C#]  
7 public static int Compare(DateTime t1, DateTime t2)
```

8 Summary

9 Returns the sort order of the two specified instances of System.DateTime.

10 Parameters

Parameter	Description
<i>t1</i>	The first System.DateTime.
<i>t2</i>	The second System.DateTime.

11

12 Return Value

13 The return value is a negative number, zero, or a positive number reflecting the sort
14 order of the two specified instances of System.DateTime. For non-zero return values,
15 the exact value returned by this method is unspecified. The following table defines the
16 return value:

Value Type	Condition
Any negative number	$t1 < t2$.
Zero	$t1 == t2$.
Any positive number	$t1 > t2$.

17

18

1 DateTime.CompareTo(System.DateTime)

2 Method

```
3 [ILAsm]  
4 .method public final hidebysig virtual int32 CompareTo(valuetype  
5 System.DateTime value)  
  
6 [C#]  
7 public int CompareTo(DateTime value)
```

8 Summary

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

11 Parameters

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

12

13 Return Value

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

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

18

19 Description

20 [Note: This method is implemented to support the
21 System.IComparable<System.DateTime> interface.]
22
23

24

1 DateTime.CompareTo(System.Object)

2 Method

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

7 Summary

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

9 Parameters

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

10

11 Return Value

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

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.

16

17 Description

18 Any instance of `System.DateTime`, regardless of its value, is considered greater than a
19 null reference.

20

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

22

23

1 **Exceptions**

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

2

3

1 DateTime.DaysInMonth(System.Int32, 2 System.Int32) Method

```
3 [ILAsm]  
4 .method public hidebysig static int32 DaysInMonth(int32 year, int32 month)  
5 [C#]  
6 public static int DaysInMonth(int year, int month)
```

7 Summary

8 Returns the number of days in a specified month of a specified year.

9 Parameters

Parameter	Description
<i>year</i>	A System.Int32 containing the year.
<i>month</i>	The month (a System.Int32 between 1 and 12).

10

11 Return Value

12 A System.Int32 set to the number of days in the specified month for the specified year.
13 If the specified month is February, the return value is 28 or 29 depending upon whether
14 the specified year is a leap year.

15 Exceptions

Exception	Condition
System.ArgumentOutOfRangeException	<i>month</i> is less than 1 or greater than 12.

16

17

1 DateTime.Equals(System.DateTime) Method

```
2 [ILAsm]  
3 .method public hidebysig virtual bool Equals(valuetype System.DateTime  
4 value)  
5 [C#]  
6 public override bool Equals(DateTime value)
```

7 Summary

8 Returns a `System.Boolean` indicating whether the current instance is equal to the
9 specified `DateTime`.

10 Parameters

Parameter	Description
<i>value</i>	A <code>System.DateTime</code> to compare with the current instance.

11

12 Return Value

13 `true` if *value* is equal to the current instance; otherwise, `false`.

14 Description

15 [Note: This method is implemented to support the
16 `System.IEquatable<System.DateTime>` interface.]
17
18

19

1 DateTime.Equals(System.Object) Method

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

6 Summary

7 Returns a System.Boolean indicating whether the current instance is equal to a
8 specified object.

9 Parameters

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

10

11 Return Value

12 true if *value* is a specified System.DateTime instance is equal to the current instance;
13 otherwise, false.

14 Description

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

18

1 DateTime.Equals(System.DateTime, 2 System.DateTime) Method

```
3 [ILAsm]  
4 .method public hidebysig static bool Equals(valuetype System.DateTime t1,  
5 valuetype System.DateTime t2)  
6 [C#]  
7 public static bool Equals(DateTime t1, DateTime t2)
```

8 Summary

9 Returns a System.Boolean indicating whether two specified instances of
10 System.DateTime are equal.

11 Parameters

Parameter	Description
<i>t1</i>	The first System.DateTime.
<i>t2</i>	The second System.DateTime.

12 13 Return Value

14 true if the two System.DateTime values are equal; otherwise, false.

15

1 DateTime.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 this 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 DateTime.IsLeapYear(System.Int32) Method

```
2 [ILAsm]  
3 .method public hidebysig static bool IsLeapYear(int32 year)  
4 [C#]  
5 public static bool IsLeapYear(int year)
```

6 Summary

7 Returns a System.Boolean value indicating whether a specified year is a leap year.

8 Parameters

Parameter	Description
<i>year</i>	A System.Int32 representing the year. <i>year</i> can be positive or negative.

9

10 Return Value

11 true if the specified year is a leap year; otherwise, false.

12

1 DateTime.op_Addition(System.DateTime, 2 System.TimeSpan) Method

```
3 [ILAsm]  
4 .method public hidebysig static specialname valuetype System.DateTime  
5 op_Addition(valuetype System.DateTime d, valuetype System.TimeSpan t)  
  
6 [C#]  
7 public static DateTime operator +(DateTime d, TimeSpan t)
```

8 Summary

9 Adds a specified System.TimeSpan value to a specified System.DateTime value.

10 Parameters

Parameter	Description
<i>d</i>	A System.DateTime value.
<i>t</i>	A System.TimeSpan value.

11 12 Return Value

13 A System.DateTime instance that is the sum of the values of *d* and *t*.

14 Description

15 The returned value is equivalent to `DateTime(d.Ticks + t.Ticks)`.

16 Exceptions

Exception	Condition
System.ArgumentOutOfRangeException	The resulting date and time is less than System.DateTime.MinValue or greater than System.DateTime.MaxValue.

17

18

1 DateTime.op_Equality(System.DateTime, 2 System.DateTime) Method

```
3 [ILAsm]  
4 .method public hidebysig static specialname bool op_Equality(valuetype  
5 System.DateTime d1, valuetype System.DateTime d2)  
  
6 [C#]  
7 public static bool operator ==(DateTime d1, DateTime d2)
```

8 Summary

9 Returns a System.Boolean value indicating whether the two specified instances of
10 System.DateTime are equal.

11 Parameters

Parameter	Description
<i>d1</i>	The first System.DateTime to compare.
<i>d2</i>	The second System.DateTime to compare.

12

13 Return Value

14 true if *d1*.Ticks value is equal to the *d2*.Ticks value; otherwise, false.

15

1 DateTime.op_GreaterThan(System.DateTime, 2 System.DateTime) Method

```
3 [ILAsm]  
4 .method public hidebysig static specialname bool op_GreaterThan(valuetype  
5 System.DateTime t1, valuetype System.DateTime t2)  
  
6 [C#]  
7 public static bool operator >(DateTime t1, DateTime t2)
```

8 Summary

9 Returns a System.Boolean value indicating whether one specified System.DateTime is
10 greater than another specified System.DateTime.

11 Parameters

Parameter	Description
<i>t1</i>	A System.DateTime.
<i>t2</i>	A System.DateTime.

12

13 Return Value

14 true if *t1*.Ticks value is greater than the *t2*.Ticks value; otherwise, false.

15

1
2 **DateTime.op_GreaterThanOrEqual(System.DateTime, System.DateTime) Method**
3

```
4 [ILAsm]  
5 .method public hidebysig static specialname bool  
6 op_GreaterThanOrEqual(valuetype System.DateTime t1, valuetype  
7 System.DateTime t2)  
  
8 [C#]  
9 public static bool operator >=(DateTime t1, DateTime t2)
```

10 **Summary**

11 Returns a *System.Boolean* value indicating whether one specified *System.DateTime* is
12 greater than or equal to another specified *System.DateTime*.

13 **Parameters**

Parameter	Description
<i>t1</i>	A <i>System.DateTime</i> .
<i>t2</i>	A <i>System.DateTime</i> .

14
15 **Return Value**

16 true if *t1.Ticks* value is greater than or equal to *t2.Ticks* value; otherwise, false.

17

1 DateTime.op_Inequality(System.DateTime, 2 System.DateTime) Method

```
3 [ILAsm]  
4 .method public hidebysig static specialname bool op_Inequality(valuetype  
5 System.DateTime d1, valuetype System.DateTime d2)  
  
6 [C#]  
7 public static bool operator !=(DateTime d1, DateTime d2)
```

8 Summary

9 Returns a System.Boolean value indicating whether two specified instances of
10 System.DateTime are not equal.

11 Parameters

Parameter	Description
<i>d1</i>	A System.DateTime.
<i>d2</i>	A System.DateTime.

12 13 Return Value

14 true if *d1*.Ticks value is not equal to *d2*.Ticks value; otherwise, false.

15

1 DateTime.op_LessThan(System.DateTime, System.DateTime) Method

```
3 [ILAsm]  
4 .method public hidebysig static specialname bool op_LessThan(valuetype  
5 System.DateTime t1, valuetype System.DateTime t2)  
  
6 [C#]  
7 public static bool operator <(DateTime t1, DateTime t2)
```

8 Summary

9 Returns a System.Boolean value indicating whether one specified System.DateTime is
10 less than another specified System.DateTime.

11 Parameters

Parameter	Description
<i>t1</i>	A System.DateTime.
<i>t2</i>	A System.DateTime.

12

13 Return Value

14 true if *t1*.Ticks value is less than *t2*.Ticks value; otherwise, false.

15

1
2 **DateTime.op_LessThanOrEqual(System.DateTime, System.DateTime) Method**
3

```
4 [ILAsm]  
5 .method public hidebysig static specialname bool  
6 op_LessThanOrEqual(valuetype System.DateTime t1, valuetype System.DateTime  
7 t2)  
8  
9 [C#]  
10 public static bool operator <=(DateTime t1, DateTime t2)
```

11 **Summary**

12 Returns a *System.Boolean* value indicating whether one specified *System.DateTime* is less than or equal to another specified *System.DateTime*.

13 **Parameters**

Parameter	Description
<i>t1</i>	A <i>System.DateTime</i> .
<i>t2</i>	A <i>System.DateTime</i> .

14
15 **Return Value**

16 true if *t1.Ticks* value is less than or equal to *t2.Ticks* value; otherwise, false.

17

1 DateTime.op_Subtraction(System.DateTime, 2 System.TimeSpan) Method

```
3 [ILAsm]  
4 .method public hidebysig static specialname valuetype System.DateTime  
5 op_Subtraction(valuetype System.DateTime d, valuetype System.TimeSpan t)  
  
6 [C#]  
7 public static DateTime operator -(DateTime d, TimeSpan t)
```

8 Summary

9 Subtracts a specified System.TimeSpan from a specified System.DateTime.

10 Parameters

Parameter	Description
<i>d</i>	A System.DateTime.
<i>t</i>	A System.TimeSpan.

11

12 Return Value

13 A System.DateTime whose value is the value of *d* minus the value of *t*.

14 Description

15 The returned value is equivalent to System.DateTime(*d*.Ticks - *t*.Ticks).

16 Exceptions

Exception	Condition
System.ArgumentOutOfRangeException	The resulting date and time is less than System.DateTime.MinValue or greater than System.DateTime.MaxValue.

17

18

1 DateTime.op_Subtraction(System.DateTime, 2 System.DateTime) Method

```
3 [ILAsm]  
4 .method public hidebysig static specialname valuetype System.TimeSpan  
5 op_Subtraction(valuetype System.DateTime d1, valuetype System.DateTime d2)  
6 [C#]  
7 public static TimeSpan operator -(DateTime d1, DateTime d2)
```

8 Summary

9 Subtracts a specified System.DateTime from another specified System.DateTime value,
10 producing a time interval.

11 Parameters

Parameter	Description
<i>d1</i>	A System.DateTime (the minuend).
<i>d2</i>	A System.DateTime (the subtrahend).

12

13 Return Value

14 A System.TimeSpan that is the time interval between *d1* and *d2*.

15 Description

16 The returned value is equivalent to System.TimeSpan(*d1*.Ticks - *d2*.Ticks).

17 Exceptions

Exception	Condition
System.ArgumentOutOfRangeException	The resulting date and time is less than System.DateTime.MinValue or greater than System.DateTime.MaxValue.

18

19

DateTime.Parse(System.String) Method

```
[ILAsm]  
.method public hidebysig static valuetype System.DateTime Parse(string s)  
  
[C#]  
public static DateTime Parse(string s)
```

Summary

Returns the specified System.String converted to a System.DateTime value.

Parameters

Parameter	Description
s	A System.String containing a value to convert. The string is interpreted using the System.Globalization.DateTimeStyles.None style.

Return Value

The System.DateTime value obtained from s.

Description

This version of System.DateTime.Parse is equivalent to System.DateTime.Parse(s, null, System.Globalization.DateTimeStyles.None).

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

In order for the string to be successfully parsed, it is required to represent a date and time value in one of the standard System.DateTime patterns described in System.Globalization.DateTimeFormatInfo.

If the string contains only a time, and no date, then the current date (System.DateTime.Now) is used. If the string contains only a date and no time, this method assumes 12 a.m.

Any leading, trailing, and inner white space characters are ignored.

Exceptions

Exception	Condition
System.ArgumentNullException	s is a null reference.

System.FormatException

s does not contain a valid string representation of a time or date and time.

1

2

DateTime.Parse(System.String, System.IFormatProvider) Method

```
[ILAsm]
.method public hidebysig static valuetype System.DateTime Parse(string s,
class System.IFormatProvider provider)

[C#]
public static DateTime Parse(string s, IFormatProvider provider)
```

Summary

Returns the specified `System.String` converted to a `System.DateTime` 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.DateTimeStyles.None</code> style.
<i>provider</i>	A <code>System.IFormatProvider</code> that supplies a <code>System.Globalization.DateTimeFormatInfo</code> object containing culture-specific format information about <i>s</i> .

Return Value

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

Description

This version of `System.DateTime.Parse` is equivalent to `System.DateTime.Parse(s, provider, System.Globalization.DateTimeStyles.None)`.

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

In order for the string to be successfully parsed, it is required to represent a date and time value in one of the standard `System.DateTime` patterns described in `System.Globalization.DateTimeFormatInfo`.

If the string contains only a time, and no date, then the current date (`System.DateTime.Now`) is used. If the string contains only a date and no time, this method assumes 12 a.m.

1

2 Any leading, trailing, and inner white space characters are ignored.

3 **Exceptions**

Exception	Condition
System.ArgumentException	s is a null reference.
System.FormatException	s does not contain a valid string representation of a time or date and time.

4

5

1 DateTime.Parse(System.String, 2 System.IFormatProvider, 3 System.Globalization.DateTimeStyles) 4 Method

```
5 [ILAsm]  
6 .method public hidebysig static valuetype System.DateTime Parse(string s,  
7 class System.IFormatProvider provider, valuetype  
8 System.Globalization.DateTimeStyles styles)  
  
9 [C#]  
10 public static DateTime Parse(string s, IFormatProvider provider,  
11 DateTimeStyles styles)
```

12 Summary

13 Returns the specified System.String converted to a System.DateTime value.

14 Parameters

Parameter	Description
<i>s</i>	A System.String containing the value to convert.
<i>provider</i>	A System.IFormatProvider that supplies a System.Globalization.DateTimeFormatInfo object containing culture-specific format information about <i>s</i> .
<i>styles</i>	One or more System.Globalization.DateTimeStyles values that specify the style of <i>s</i> . Specify multiple values for <i>styles</i> using the bitwise OR operator.

15 16 Return Value

17 The System.DateTime value obtained from *s*.

18 Description

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

23
24 In order for the string to be successfully parsed, it is required to represent a date and
25 time value in one of the standard System.DateTime patterns described in
26 System.Globalization.DateTimeFormatInfo.

1
2 If the string contains only a time, and no date, and if the *styles* parameter is set to
3 `System.Globalization.DateTimeStyles.NoCurrentDateDefault` the Gregorian year 1,
4 month 1, day 1 are used. In all other cases where a date is not specified, the current
5 date (`System.DateTime.Now`) is used.

6
7 If the string contains only a date and no time, this method assumes 12 a.m.

8
9 For all settings of the *styles* parameter, any leading, trailing, and inner white space
10 characters are ignored.

11 Exceptions

Exception	Condition
System.ArgumentException	s is a null reference.
System.FormatException	s does not contain a valid string representation of a time or date and time.

12

13

DateTime.ParseExact(System.String, System.String, System.IFormatProvider, System.Globalization.DateTimeStyles)

Method

```
[ILAsm]
.method public hidebysig static valuetype System.DateTime
ParseExact(string s, string format, class System.IFormatProvider provider,
valuetype System.Globalization.DateTimeStyles style)

[C#]
public static DateTime ParseExact(string s, string format, IFormatProvider
provider, DateTimeStyles style)
```

Summary

Converts the `System.String` representation of a date and time to its `System.DateTime` equivalent using a specified style, the expected format, and culture-specific format information.

Parameters

Parameter	Description
<i>s</i>	A <code>System.String</code> containing a date and time to convert. The format of the string is required to match the specified format exactly.
<i>format</i>	A <code>System.String</code> containing the expected format of <i>s</i> . [<i>Note:</i> For a list of valid <i>format</i> values, see <code>System.Globalization.DateTimeFormatInfo</code> .]
<i>provider</i>	A <code>System.IFormatProvider</code> that supplies a <code>System.Globalization.DateTimeFormatInfo</code> object containing culture-specific format information about <i>s</i> .
<i>style</i>	One or more <code>System.Globalization.DateTimeStyles</code> values that specify the style of <i>s</i> . Specify multiple values for <i>styles</i> using the bitwise OR operator.

Return Value

A `System.DateTime` equivalent to the date and time contained in *s*.

Description

1 `System.DateTime.ParseExact` constructs a `System.DateTime` from the string `s`. The
 2 string is required to specify a date and, optionally, a time in the provided format.
 3
 4 The string `s` is parsed using the culture-specific formatting information from the
 5 `System.Globalization.DateTimeFormatInfo` instance supplied by *provider*. If *provider*
 6 is null or a `System.Globalization.DateTimeFormatInfo` cannot be obtained from
 7 *provider*, the formatting information for the current system culture is used.
 8
 9 If the `s` string contains only a time, and no date, and if the *styles* parameter is set to
 10 `System.Globalization.DateTimeStyles.NoCurrentDateDefault` the Gregorian year 1,
 11 month 1, day 1 are used, and no leading, trailing, or inner white space characters are
 12 allowed. In all other cases where a date is not specified, the current date
 13 (`System.DateTime.Now`) is used.
 14
 15 If the `s` string contains only a date and no time, this method assumes 12 a.m.
 16
 17 [Note: For information on formatting system-supplied data types, see the
 18 `System.IFormattable` interface.]
 19
 20

21 Exceptions

Exception	Condition
System.ArgumentNullException	<code>s</code> or <code>format</code> is a null reference.
System.FormatException	<p><code>s</code> or <code>format</code> is an empty string.</p> <p>-or-</p> <p><code>s</code> does not contain a date and time that were recognized as one of the patterns specified in <code>format</code>.</p>

22

23 Example

24 This example demonstrates the `System.DateTime.ParseExact` method.

25
 26 [C#]

```

27 using System;
28 using System.Globalization;
29
30 public class DateTimeTest {
31     public static void Main() {
32         DateTimeFormatInfo dtfi = new DateTimeFormatInfo();
33
34         DateTime dt = DateTime.ParseExact(" January 22 ", dtfi.MonthDayPattern,
35     null, DateTimeStyles.AllowWhiteSpaces);
36         Console.WriteLine(dt);
  
```

```
1  }
2  }
3
4  The output is
5
6  1/22/2001 12:00:00 AM
7
8
```

DateTime.ParseExact(System.String, System.String[], System.IFormatProvider, System.Globalization.DateTimeStyles) Method

```
[ILAsm]
.method public hidebysig static valuetype System.DateTime
ParseExact(string s, string[] formats, class System.IFormatProvider
provider, valuetype System.Globalization.DateTimeStyles style)

[C#]
public static DateTime ParseExact(string s, string[] formats,
IFormatProvider provider, DateTimeStyles style)
```

Summary

Converts the `System.String` representation of a date and time to its `System.DateTime` equivalent using a specified style, an array of expected formats, and culture-specific format information.

Parameters

Parameter	Description
<i>s</i>	A <code>System.String</code> containing one or more dates and times to convert. The format of the string is required to match the specified format exactly.
<i>formats</i>	A <code>System.String</code> array containing the expected formats of <i>s</i> . [<i>Note:</i> For a list of valid <i>format</i> values, see <code>System.Globalization.DateTimeFormatInfo</code> .]
<i>provider</i>	A <code>System.IFormatProvider</code> that supplies a <code>System.Globalization.DateTimeFormatInfo</code> object containing culture-specific format information about <i>s</i> .
<i>style</i>	One or more <code>System.Globalization.DateTimeStyles</code> values that specify the style of <i>s</i> . Specify multiple values for <i>styles</i> using the bitwise OR operator.

Return Value

A `System.DateTime` equivalent to the date and time contained in *s*.

Description

`System.DateTime.ParseExact` constructs a `System.DateTime` from the `System.String`. The string is required to specify a date and, optionally, a time in the

1 provided format.

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

7
8 If the *s* string contains only a time, and no date, and if the *styles* parameter is set to
9 `System.Globalization.DateTimeStyles.NoCurrentDateDefault` the Gregorian year 1,
10 month 1, day 1 are used, and no leading, trailing, or inner white space characters are
11 allowed. In all other cases where a date is not specified, the current date
12 (`System.DateTime.Now`) is used.

13
14 If the *s* string contains only a date and no time, this method assumes 12 a.m.

15
16 [Note: For information on formatting system-supplied data types, see the
17 `System.IFormattable` interface.]

20 Exceptions

Exception	Condition
System.ArgumentNullException	<i>s</i> or <i>formats</i> is a null reference.
System.FormatException	<i>s</i> or <i>format</i> is an empty string.
	-or- <i>s</i> does not contain a date and time that were recognized as the pattern specified in <i>format</i>

22 Example

23 This example demonstrates the `System.DateTime.ParseExact` method.

24 [C#]

```
25  
26 using System;  
27 using System.Globalization;  
28  
29 public class DateTimeTest {  
30     public static void Main() {  
31         DateTimeFormatInfo dtfi = new DateTimeFormatInfo();  
32         string [] patterns = {dtfi.LongTimePattern, dtfi.ShortTimePattern};  
33  
34         DateTime dt = DateTime.ParseExact("10:11:12", patterns, null,  
35         DateTimeStyles.NoCurrentDateDefault);  
36         Console.WriteLine(dt);
```

```
1  }
2  }
3
4  The output is
5
6  1/1/0001 10:11:12 AM
7
```

1 DateTime.ParseExact(System.String, 2 System.String, System.IFormatProvider) 3 Method

```
4 [ILAsm]  
5 .method public hidebysig static valuetype System.DateTime  
6 ParseExact(string s, string format, class System.IFormatProvider provider)  
  
7 [C#]  
8 public static DateTime ParseExact(string s, string format, IFormatProvider  
9 provider)
```

10 Summary

11 Converts the specified System.String representation of a date and time to its
12 System.DateTime equivalent using a specified format and System.IFormatProvider.

13 Parameters

Parameter	Description
<i>s</i>	A System.String containing a date and time to convert. The format of the string is required to match the specified format exactly.
<i>format</i>	A System.String containing the expected format of <i>s</i> . [<i>Note:</i> For a list of valid <i>format</i> values, see System.Globalization.DateTimeFormatInfo.]
<i>provider</i>	A System.IFormatProvider that supplies a System.Globalization.DateTimeFormatInfo object containing culture-specific format information about <i>s</i> .

14 15 Return Value

16 A System.DateTime equivalent to the date and time contained in *s*.

17 Description

18 System.DateTime.ParseExact constructs a System.DateTime from the string *s*. The
19 string is required to specify a date and, optionally, a time in the specified format.

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

25
26 If the *s* string contains only a time, and no date, then the current date

1 (System.DateTime.Now) is used. If the string contains only a date and no time, this
2 method assumes 12 a.m.

3
4 Leading, trailing, and inner white space characters are not allowed.

5
6 [Note: For information on formatting system-supplied data types, see the
7 System.IFormattable interface.]
8
9

10 Exceptions

Exception	Condition
System.ArgumentNullException	<i>s</i> or <i>format</i> is a null reference.
System.FormatException	<i>s</i> or <i>format</i> is an empty string. -or- <i>s</i> does not contain a date and time that were recognized as the pattern specified in <i>format</i> .

11 12 Example

13 This example demonstrates the System.DateTime.ParseExact method.

```
14 [C#]  
15  
16 using System;  
17 using System.Globalization;  
18  
19 public class DateTimeTest {  
20     public static void Main() {  
21         DateTimeFormatInfo dtfi = new DateTimeFormatInfo();  
22  
23         DateTime dt = DateTime.ParseExact("January 22", dtfi.MonthDayPattern,  
24 null);  
25         Console.WriteLine(dt);  
26     }  
27 }  
28
```

29 The output is

```
30  
31 1/22/2001 12:00:00 AM  
32
```

1 DateTime.Subtract(System.TimeSpan)

2 Method

```
3 [ILAsm]  
4 .method public hidebysig instance valuetype System.DateTime  
5 Subtract(valuetype System.TimeSpan value)  
  
6 [C#]  
7 public DateTime Subtract(TimeSpan value)
```

8 Summary

9 Subtracts a specified `System.TimeSpan` from the current instance.

10 Parameters

Parameter	Description
<i>value</i>	An instance of <code>System.TimeSpan</code> .

11

12 Return Value

13 A new `System.DateTime` instance equal to the date and time represented by the current
14 instance minus the time interval of the specified `System.TimeSpan`.

15 Exceptions

Exception	Condition
System.ArgumentOutOfRangeException	The resulting date and time is less than <code>System.DateTime.MinValue</code> or greater than <code>System.DateTime.MaxValue</code> .

16

17

1 DateTime.Subtract(System.DateTime)

2 Method

```
3 [ILAsm]  
4 .method public hidebysig instance valuetype System.TimeSpan  
5 Subtract(valuetype System.DateTime value)  
6 [C#]  
7 public TimeSpan Subtract(DateTime value)
```

8 Summary

9 Subtracts a specified date and time from the current instance.

10 Parameters

Parameter	Description
<i>value</i>	An instance of <code>System.DateTime</code> .

11

12 Return Value

13 A `System.TimeSpan` interval equal to the date and time represented by the current
14 instance minus the date and time represented by the specified `System.DateTime`.

15

1 DateTime.ToLocalTime() Method

```
2 [ILAsm]  
3 .method public hidebysig instance valuetype System.DateTime ToLocalTime()  
4 [C#]  
5 public DateTime ToLocalTime()
```

6 Summary

7 Converts the universal time coordinate (UTC) time value in the current instance to local
8 time.

9 Return Value

10 An instance of `System.DateTime` equivalent of the time value in the current instance,
11 adjusted to the local time zone and daylight saving time. If the result is too large or too
12 small to be represented as a `System.DateTime`, this method returns a `System.DateTime`
13 set to `System.DateTime.MaxValue` or `System.DateTime.MinValue`.

14 Description

15 This method assumes that the current instance of `System.DateTime` holds the UTC time
16 value, and not a local time. Each time it is invoked, this method performs the necessary
17 modifications on the `System.DateTime` to derive the local time, whether the current
18 `System.DateTime` holds the UTC time or not.

19
20 The local time zone information is obtained from the operating system.

21

1 DateTime.ToLongDateString() Method

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

6 Summary

7 Converts the date denoted by the current instance to its equivalent long date
8 System.String representation.

9 Return Value

10 A System.String containing the same value as a System.String returned by
11 System.DateTime.ToString ("D", null).

12 Description

13 The value of the current instance is formatted using the long date format specifier, 'D'.
14

15 [*Note:* This format uses the culture of the current thread. To specify formatting using a
16 different culture, use System.DateTime.ToString.

17
18 For more information regarding the long date specifier, see
19 System.Globalization.DateTimeFormatInfo.
20

21]
22

1 DateTime.ToLongTimeString() Method

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

6 Summary

7 Converts the time denoted by the current instance to its equivalent long time
8 System.String representation.

9 Return Value

10 A System.String containing the same value as a System.String returned by
11 System.DateTime.ToString ("T", null).

12 Description

13 The value of the current instance is formatted using the long time format specifier, 'T'.
14

15 [*Note:* This format uses the culture of the current thread. To specify formatting using a
16 different culture, use System.DateTime.ToString.

17
18 For more information regarding the long time specifier, see
19 System.Globalization.DateTimeFormatInfo.
20

21]
22

1 DateTime.ToShortDateString() Method

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

6 Summary

7 Converts the date denoted by the current instance to its equivalent short date
8 System.String representation.

9 Return Value

10 A System.String containing the same value as a System.String returned by
11 System.DateTime.ToString ("d", null).

12 Description

13 The value of the current instance is formatted using the short date format specifier, 'd'.
14

15 [*Note:* This format uses the culture of the current thread. To specify formatting using a
16 different culture, use System.DateTime.ToString.

17
18 For more information regarding the short date specifier, see
19 System.Globalization.DateTimeFormatInfo.
20

21]
22

1 DateTime.ToShortTimeString() Method

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

6 Summary

7 Converts the time denoted by this instance to its equivalent short time `System.String`
8 representation.

9 Return Value

10 A `System.String` containing the same value as a `System.String` returned by
11 `System.DateTime.ToString ("t", null)`.

12 Description

13 The value of the current instance is formatted using the short time format specifier, 't'.
14

15 [*Note:* This format uses the culture of the current thread. To specify formatting using a
16 different culture, use `System.DateTime.ToString`.

17
18 For more information regarding the short time specifier, see
19 `System.Globalization.DateTimeFormatInfo`.

20
21]

22

1 DateTime.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.Globalization.DateTimeFormatInfo</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 version of `System.DateTime.ToString` is equivalent to `System.DateTime.ToString`
15 (*format*, null).

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

18
19 [*Note:* This method uses the culture information of the current thread.

20
21 For information on formatting system-supplied data types, see the
22 `System.IFormattable` interface.

23
24]

25 Exceptions

Exception	Condition
System.FormatException	The length of the <i>format</i> string is 1, and it is not one of the format specifier characters defined for <code>System.Globalization.DateTimeFormatInfo</code> . -or-

	The <i>format</i> string does not contain a valid custom format pattern.
--	--

1

2

1 DateTime.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.DateTime.ToString` is equivalent to `System.DateTime.ToString`
13 ("`G`", `null`).

14
15 [*Note:* For more information about the general format specifier ("`G`") see
16 `System.Globalization.DateTimeFormatInfo`.

17 This method overrides `System.Object.ToString`.

18]
19
20]

21

1 DateTime.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. [Note: For a list of valid values, see <code>System.Globalization.DateTimeFormatInfo</code> .]
<i>provider</i>	A <code>System.IFormatProvider</code> that supplies a <code>System.Globalization.DateTimeFormatInfo</code> instance containing culture-specific formatting information.

11 12 Return Value

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

16 Description

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

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

21 [Note: For more information regarding the standard format specifier, see
22 `System.Globalization.DateTimeFormatInfo`. For information on formatting system-
23 supplied data types, see the `System.IFormattable` interface.

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

25]
26
27
28
29

1 Exceptions

Exception	Condition
System.FormatException	The length of the <i>format</i> string is 1, and it is not one of the format specifier characters defined for <code>System.Globalization.DateTimeFormatInfo</code> . -or- The <i>format</i> string does not contain a valid custom format pattern.

2

3

DateTime.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.DateTimeFormatInfo</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 formatting information in the `System.Globalization.DateTimeFormatInfo` instance supplied by *provider*.

Description

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

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

[*Note:* The general format specifier ("G") provides the general date pattern including the long time form, equivalent to `System.Globalization.DateTimeFormatInfo.ShortDatePattern` combined with `System.Globalization.DateTimeFormatInfo.LongTimePattern`. For more information on format specifiers, see `System.Globalization.DateTimeFormatInfo`. For information on formatting system-supplied data types, see the `System.IFormattable` interface.

]

1 DateTime.ToUniversalTime() Method

```
2 [ILAsm]  
3 .method public hidebysig instance valuetype System.DateTime  
4 ToUniversalTime()  
5 [C#]  
6 public DateTime ToUniversalTime()
```

7 Summary

8 Converts the current `System.DateTime` value to coordinated universal time (UTC).

9 Return Value

10 The UTC `System.DateTime` equivalent of the current `System.DateTime` value. If the
11 result is too large or too small to be represented as a `System.DateTime`, the current
12 function returns a `System.DateTime` set to `System.DateTime.MaxValue` or
13 `System.DateTime.MinValue`.

14 Description

15 This method assumes that the current instance of `System.DateTime` holds the local time
16 value, and not a UTC time. Therefore each time it is run, this method performs the
17 necessary modifications on the `System.DateTime` to derive the UTC time, whether the
18 current `System.DateTime` holds the local time or not.

19
20 The local time zone information is obtained from the operating system.

21

1 DateTime.Date Property

```
2 [ILAsm]  
3 .property valuetype System.DateTime Date { public hidebysig specialname  
4 instance valuetype System.DateTime get_Date() }  
5 [C#]  
6 public DateTime Date { get; }
```

7 Summary

8 Gets the date component of the current instance.

9 Property Value

10 A new *System.DateTime* instance with the same date as the current instance, and the
11 time value set to midnight (00:00:00).

12 Description

13 This property is read-only.

14

1 DateTime.Day Property

```
2 [ILAsm]  
3 .property int32 Day { public hidebysig specialname instance int32  
4 get_Day() }  
  
5 [C#]  
6 public int Day { get; }
```

7 Summary

8 Gets the day of the month represented by the current instance.

9 Property Value

10 A `System.Int32` between 1 and 31 set to the day of the month component of the
11 current instance.

12 Description

13 This property is read-only.

14

1 DateTime.DayOfYear Property

```
2 [ILAsm]  
3 .property int32 DayOfYear { public hidebysig specialname instance int32  
4 get_DayOfYear() }  
5 [C#]  
6 public int DayOfYear { get; }
```

7 Summary

8 Gets the day of the year represented by the current instance.

9 Property Value

10 A `System.Int32` between 1 and 366 set to the day of the year component of the current
11 instance.

12 Description

13 This property is read-only.

14

1 DateTime.Hour Property

```
2 [ILAsm]  
3 .property int32 Hour { public hidebysig specialname instance int32  
4 get_Hour() }  
5 [C#]  
6 public int Hour { get; }
```

7 Summary

8 Gets the hour represented by the current instance.

9 Property Value

10 A `System.Int32` between 0 and 23 set to the hour component of the current instance.

11 Description

12 This property is read-only.

13

1 DateTime.Millisecond Property

```
2 [ILAsm]  
3 .property int32 Millisecond { public hidebysig specialname instance int32  
4 get_Millisecond() }  
  
5 [C#]  
6 public int Millisecond { get; }
```

7 Summary

8 Gets the milliseconds component of the date represented by the current instance.

9 Property Value

10 A `System.Int32` between 0 and 999 set to the milliseconds component of the current
11 instance.

12 Description

13 This property is read-only.

14

1 DateTime.Minute Property

```
2 [ILAsm]  
3 .property int32 Minute { public hidebysig specialname instance int32  
4 get_Minute() }  
5 [C#]  
6 public int Minute { get; }
```

7 Summary

8 Gets the minute component of the date represented by the current instance.

9 Property Value

10 A `System.Int32` between 0 and 59 set to the minute component of the current instance.

11 Description

12 This property is read-only.

13

1 DateTime.Month Property

```
2 [ILAsm]  
3 .property int32 Month { public hidebysig specialname instance int32  
4 get_Month() }  
5 [C#]  
6 public int Month { get; }
```

7 Summary

8 Gets the month component of the date represented by the current instance.

9 Property Value

10 A `System.Int32` between 1 and 12 set to the month component of the current instance.

11 Description

12 This property is read-only.

13

1 DateTime.Now Property

```
2 [ILAsm]  
3 .property valuetype System.DateTime Now { public hidebysig static  
4 specialname valuetype System.DateTime get_Now() }  
5 [C#]  
6 public static DateTime Now { get; }
```

7 Summary

8 Gets a `System.DateTime` representing the current local date and time.

9 Description

10 The resolution of this property depends on the system timer.

11

12 This property is read-only.

13

1 DateTime.Second Property

```
2 [ILAsm]  
3 .property int32 Second { public hidebysig specialname instance int32  
4 get_Second() }  
  
5 [C#]  
6 public int Second { get; }
```

7 Summary

8 Gets the seconds component of the date represented by the current instance.

9 Property Value

10 A `System.Int32` between 0 and 59 set to the seconds component of the current
11 instance.

12 Description

13 This property is read-only.

14

1 DateTime.Ticks Property

```
2 [ILAsm]  
3 .property int64 Ticks { public hidebysig specialname instance int64  
4 get_Ticks() }  
5 [C#]  
6 public long Ticks { get; }
```

7 Summary

8 Gets the number of 100-nanosecond ticks that represent the date and time of the
9 current instance.

10 Property Value

11 A `System.Int64` set to the number of ticks that represent the date and time of the
12 current instance.

13 Description

14 The value of this property is the number of 100-nanosecond intervals that have elapsed
15 since 00:00:00, 1/1/0001. The value of the property is between
16 `System.DateTime.MinValue` and `System.DateTime.MaxValue`.

17
18 This property is read-only.

19

1 DateTime.TimeOfDay Property

```
2 [ILAsm]  
3 .property valuetype System.TimeSpan TimeOfDay { public hideby sig  
4 specialname instance valuetype System.TimeSpan get_TimeOfDay() }  
  
5 [C#]  
6 public TimeSpan TimeOfDay { get; }
```

7 Summary

8 Gets the time of day of the current instance.

9 Property Value

10 A `System.TimeSpan` instance set to the time component of the current instance.

11 Description

12 This property is read-only.

13

1 DateTime.Today Property

```
2 [ILAsm]  
3 .property valuetype System.DateTime Today { public hidebysig static  
4 specialname valuetype System.DateTime get_Today() }  
5 [C#]  
6 public static DateTime Today { get; }
```

7 Summary

8 Gets the current date.

9 Property Value

10 A `System.DateTime` instance set to the date of the current instance, with the time set to
11 00:00:00.

12 Description

13 This property is read-only.

14

1 DateTime.UtcNow Property

```
2 [ILAsm]  
3 .property valuetype System.DateTime UtcNow { public hidebysig static  
4 specialname valuetype System.DateTime get_UtcNow() }  
5 [C#]  
6 public static DateTime UtcNow { get; }
```

7 Summary

8 Gets the current time converted to coordinated universal time (UTC).

9 Property Value

10 A System.DateTime instance set to the current date and time in coordinated universal
11 time (UTC).

12 Description

13 This property is read-only.

14

1 DateTime.Year Property

```
2 [ILAsm]  
3 .property int32 Year { public hidebysig specialname instance int32  
4 get_Year() }  
5 [C#]  
6 public int Year { get; }
```

7 Summary

8 Gets the year component of the date represented by the current instance.

9 Property Value

10 A System.Int32 between 1 and 9999 set to the year component of the current instance.

11 Description

12 This property is read-only.

13