String Manipulation Summary (Visual Basic)

Visual Studio 2015

Visual Basic language keywords and run-time library members are organized by purpose and use.

Action	Language element
Compare two strings.	StrComp
Convert strings.	StrConv
Reverse a string.	InStrRev, StrReverse
Convert to lowercase or uppercase.	Format, LCase, UCase
Create a string of repeating characters.	Space, StrDup
Find the length of a string.	Len
Format a string.	Format, FormatCurrency, FormatDateTime, FormatNumber, FormatPercent
Manipulate strings.	InStr, Left, LTrim, Mid, Right, RTrim, Trim
Set string comparison rules.	Option Compare
Work with ASCII and ANSI values.	Asc, AscW, Chr, ChrW
Replace a specified substring.	Replace
Return a filter-based string array.	Filter
Return a specified number of substrings.	Split, Join

See Also

Keywords (Visual Basic)
Visual Basic Runtime Library Members

Strings Class

.NET Framework (current version)

The **Strings** module contains procedures used to perform string operations.

Namespace: Microsoft.VisualBasic
Assembly: Microsoft.VisualBasic (in Microsoft.VisualBasic.dll)

Inheritance Hierarchy

System.Object
  Microsoft.VisualBasic.Strings

Syntax

VB

```
<StandardModuleAttribute>
Public NotInheritable Class Strings
```

Methods

	Name	Description
S	Asc(Char)	Returns an **Integer** value representing the character code corresponding to a character.
S	Asc(String)	Returns an **Integer** value representing the character code corresponding to a character.
S	AscW(Char)	Returns an **Integer** value representing the character code corresponding to a character.
S	AscW(String)	Returns an **Integer** value representing the character code corresponding to a character.
S	Chr(Int32)	Returns the character associated with the specified character code.

S	ChrW(Int32)	Returns the character associated with the specified character code.
	Equals(Object)	Determines whether the specified object is equal to the current object.(Inherited from Object.)
S	Filter(Object(), String, Boolean, CompareMethod)	Returns a zero-based array containing a subset of a **String** array based on specified filter criteria.
S	Filter(String(), String, Boolean, CompareMethod)	Returns a zero-based array containing a subset of a **String** array based on specified filter criteria.
S	Format(Object, String)	Returns a string formatted according to instructions contained in a format **String** expression.
S	FormatCurrency(Object, Int32, TriState, TriState, TriState)	Returns an expression formatted as a currency value using the currency symbol defined in the system control panel.
S	FormatDateTime(DateTime, DateFormat)	Returns a string expression representing a date/time value.
S	FormatNumber(Object, Int32, TriState, TriState, TriState)	Returns an expression formatted as a number.
S	FormatPercent(Object, Int32, TriState, TriState, TriState)	Returns an expression formatted as a percentage (that is, multiplied by 100) with a trailing % character.
S	GetChar(String, Int32)	Returns a **Char** value representing the character from the specified index in the supplied string.
	GetHashCode()	Serves as the default hash function. (Inherited from Object.)
	GetType()	Gets the Type of the current instance.(Inherited from Object.)
S	InStr(Int32, String, String, CompareMethod)	Returns an integer specifying the start position of the first occurrence of one string within another.
S	InStr(String, String, CompareMethod)	Returns an integer specifying the start position of the first occurrence of one string within another.
S	InStrRev(String, String, Int32, CompareMethod)	Returns the position of the first occurrence of one string within another, starting from the right side of the string.
S	Join(Object(), String)	Returns a string created by joining a number of substrings contained in an array.
S	Join(String(), String)	Returns a string created by joining a number of substrings contained in an array.

S	LCase(Char)	Returns a string or character converted to lowercase.
S	LCase(String)	Returns a string or character converted to lowercase.
S	Left(String, Int32)	Returns a string containing a specified number of characters from the left side of a string.
S	Len(Boolean)	Returns an integer containing either the number of characters in a string or the nominal number of bytes required to store a variable.
S	Len(Byte)	Returns an integer containing either the number of characters in a string or the nominal number of bytes required to store a variable.
S	Len(Char)	Returns an integer containing either the number of characters in a string or the nominal number of bytes required to store a variable.
S	Len(DateTime)	Returns an integer containing either the number of characters in a string or the nominal number of bytes required to store a variable.
S	Len(Decimal)	Returns an integer containing either the number of characters in a string or the nominal number of bytes required to store a variable.
S	Len(Double)	Returns an integer containing either the number of characters in a string or the nominal number of bytes required to store a variable.
S	Len(Int16)	Returns an integer containing either the number of characters in a string or the nominal number of bytes required to store a variable.
S	Len(Int32)	Returns an integer containing either the number of characters in a string or the nominal number of bytes required to store a variable.
S	Len(Int64)	Returns an integer containing either the number of characters in a string or the nominal number of bytes required to store a variable.
S	Len(Object)	Returns an integer containing either the number of characters in a string or the nominal number of bytes required to store a variable.
S	Len(SByte)	Returns an integer containing either the number of characters in a string or the nominal number of bytes

		required to store a variable.
S	Len(Single)	Returns an integer containing either the number of characters in a string or the nominal number of bytes required to store a variable.
S	Len(String)	Returns an integer containing either the number of characters in a string or the nominal number of bytes required to store a variable.
S	Len(UInt16)	Returns an integer containing either the number of characters in a string or the nominal number of bytes required to store a variable.
S	Len(UInt32)	Returns an integer containing either the number of characters in a string or the nominal number of bytes required to store a variable.
S	Len(UInt64)	Returns an integer containing either the number of characters in a string or the nominal number of bytes required to store a variable.
S	LSet(String, Int32)	Returns a left-aligned string containing the specified string adjusted to the specified length.
S	LTrim(String)	Returns a string containing a copy of a specified string with no leading spaces (**LTrim**), no trailing spaces (**RTrim**), or no leading or trailing spaces (**Trim**).
S	Mid(String, Int32)	Returns a string that contains all the characters starting from a specified position in a string.
S	Mid(String, Int32, Int32)	Returns a string that contains a specified number of characters starting from a specified position in a string.
S	Replace(String, String, String, Int32, Int32, CompareMethod)	Returns a string in which a specified substring has been replaced with another substring a specified number of times.
S	Right(String, Int32)	Returns a string containing a specified number of characters from the right side of a string.
S	RSet(String, Int32)	Returns a right-aligned string containing the specified string adjusted to the specified length.
S	RTrim(String)	Returns a string containing a copy of a specified string with no leading spaces (**LTrim**), no trailing spaces (**RTrim**), or no leading or trailing spaces (**Trim**).

S	Space(Int32)	Returns a string consisting of the specified number of spaces.
S	Split(String, String, Int32, CompareMethod)	Returns a zero-based, one-dimensional array containing a specified number of substrings.
S	StrComp(String, String, CompareMethod)	Returns -1, 0, or 1, based on the result of a string comparison.
S	StrConv(String, VbStrConv, Int32)	Returns a string converted as specified.
S	StrDup(Int32, Char)	Returns a string or object consisting of the specified character repeated the specified number of times.
S	StrDup(Int32, Object)	Returns a string or object consisting of the specified character repeated the specified number of times.
S	StrDup(Int32, String)	Returns a string or object consisting of the specified character repeated the specified number of times.
S	StrReverse(String)	Returns a string in which the character order of a specified string is reversed.
	ToString()	Returns a string that represents the current object. (Inherited from Object.)
S	Trim(String)	Returns a string containing a copy of a specified string with no leading spaces (**LTrim**), no trailing spaces (**RTrim**), or no leading or trailing spaces (**Trim**).
S	UCase(Char)	Returns a string or character containing the specified string converted to uppercase.
S	UCase(String)	Returns a string or character containing the specified string converted to uppercase.

Remarks

This module supports the Visual Basic language keywords and run-time library members that manipulate strings.

Examples

The following example demonstrates how to split a string at its spaces.

VB

```
Dim TestString As String = "Look at these!"
' Returns an array containing "Look", "at", and "these!".
Dim TestArray() As String = Split(TestString)
```

Version Information

Universal Windows Platform
Available since 8
.NET Framework
Available since 1.1
Portable Class Library
Supported in: portable .NET platforms
Silverlight
Available since 2.0
Windows Phone
Available since 8.1

Thread Safety

Any public static (**Shared** in Visual Basic) members of this type are thread safe. Any instance members are not guaranteed to be thread safe.

See Also

Microsoft.VisualBasic Namespace
String Manipulation Summary (Visual Basic)
Keywords (Visual Basic)
Visual Basic Runtime Library Members

Return to top

Standard Date and Time Format Strings

.NET Framework (current version)

A standard date and time format string uses a single format specifier to define the text representation of a date and time value. Any date and time format string that contains more than one character, including white space, is interpreted as a custom date and time format string; for more information, see Custom Date and Time Format Strings. A standard or custom format string can be used in two ways:

- To define the string that results from a formatting operation.
- To define the text representation of a date and time value that can be converted to a DateTime or DateTimeOffset value by a parsing operation.

Standard date and time format strings can be used with both DateTime and DateTimeOffset values.

Tip

You can download the Formatting Utility, an application that enables you to apply format strings to either numeric or date and time values and displays the result string.

The following table describes the standard date and time format specifiers. Unless otherwise noted, a particular standard date and time format specifier produces an identical string representation regardless of whether it is used with a DateTime or a DateTimeOffset value. See the Notes section for additional information about using standard date and time format strings.

Format specifier	Description	Examples
"d"	Short date pattern. More information:The Short Date ("d") Format Specifier.	2009-06-15T13:45:30 -> 6/15/2009 (en-US) 2009-06-15T13:45:30 -> 15/06/2009 (fr-FR) 2009-06-15T13:45:30 -> 2009/06/15 (ja-JP)
"D"	Long date pattern. More information:The Long Date ("D") Format Specifier.	2009-06-15T13:45:30 -> Monday, June 15, 2009 (en-US) 2009-06-15T13:45:30 -> 15 июня 2009 г. (ru-RU) 2009-06-15T13:45:30 -> Montag, 15. Juni 2009 (de-DE)
"f"	Full date/time pattern (short time). More information: The Full Date Short Time ("f") Format Specifier.	2009-06-15T13:45:30 -> Monday, June 15, 2009 1:45 PM (en-US)

		2009-06-15T13:45:30 -> den 15 juni 2009 13:45 (sv-SE) 2009-06-15T13:45:30 -> Δευτέρα, 15 Ιουνίου 2009 1:45 μμ (el-GR)
"F"	Full date/time pattern (long time). More information: The Full Date Long Time ("F") Format Specifier.	2009-06-15T13:45:30 -> Monday, June 15, 2009 1:45:30 PM (en-US) 2009-06-15T13:45:30 -> den 15 juni 2009 13:45:30 (sv-SE) 2009-06-15T13:45:30 -> Δευτέρα, 15 Ιουνίου 2009 1:45:30 μμ (el-GR)
"g"	General date/time pattern (short time). More information: The General Date Short Time ("g") Format Specifier.	2009-06-15T13:45:30 -> 6/15/2009 1:45 PM (en-US) 2009-06-15T13:45:30 -> 15/06/2009 13:45 (es-ES) 2009-06-15T13:45:30 -> 2009/6/15 13:45 (zh-CN)
"G"	General date/time pattern (long time). More information: The General Date Long Time ("G") Format Specifier.	2009-06-15T13:45:30 -> 6/15/2009 1:45:30 PM (en-US) 2009-06-15T13:45:30 -> 15/06/2009 13:45:30 (es-ES) 2009-06-15T13:45:30 -> 2009/6/15 13:45:30 (zh-CN)
"M", "m"	Month/day pattern. More information: The Month ("M", "m") Format Specifier.	2009-06-15T13:45:30 -> June 15 (en-US) 2009-06-15T13:45:30 -> 15. juni (da-DK) 2009-06-15T13:45:30 -> 15 Juni (id-ID)
"O", "o"	Round-trip date/time pattern. More information: The Round-trip ("O", "o") Format Specifier.	DateTime values: 2009-06-15T13:45:30 (DateTimeKind.Local) --> 2009-06-15T13:45:30.0000000-07:00 2009-06-15T13:45:30 (DateTimeKind.Utc) --> 2009-06-15T13:45:30.0000000Z 2009-06-15T13:45:30 (DateTimeKind.Unspecified) --> 2009-06-15T13:45:30.0000000 DateTimeOffset values: 2009-06-15T13:45:30-07:00 --> 2009-06-15T13:45:30.0000000-07:00
"R", "r"	RFC1123 pattern. More information: The RFC1123 ("R", "r") Format Specifier.	2009-06-15T13:45:30 -> Mon, 15 Jun 2009 20:45:30 GMT

"s"	Sortable date/time pattern. More information: The Sortable ("s") Format Specifier.	2009-06-15T13:45:30 (DateTimeKind.Local) -> 2009-06-15T13:45:30 2009-06-15T13:45:30 (DateTimeKind.Utc) -> 2009-06-15T13:45:30
"t"	Short time pattern. More information: The Short Time ("t") Format Specifier.	2009-06-15T13:45:30 -> 1:45 PM (en-US) 2009-06-15T13:45:30 -> 13:45 (hr-HR) 2009-06-15T13:45:30 -> 01:45 م (ar-EG)
"T"	Long time pattern. More information: The Long Time ("T") Format Specifier.	2009-06-15T13:45:30 -> 1:45:30 PM (en-US) 2009-06-15T13:45:30 -> 13:45:30 (hr-HR) 2009-06-15T13:45:30 -> 01:45:30 م (ar-EG)
"u"	Universal sortable date/time pattern. More information: The Universal Sortable ("u") Format Specifier.	With a DateTime value: 2009-06-15T13:45:30 -> 2009-06-15 13:45:30Z With a DateTimeOffset value: 2009-06-15T13:45:30 -> 2009-06-15 20:45:30Z
"U"	Universal full date/time pattern. More information: The Universal Full ("U") Format Specifier.	2009-06-15T13:45:30 -> Monday, June 15, 2009 8:45:30 PM (en-US) 2009-06-15T13:45:30 -> den 15 juni 2009 20:45:30 (sv-SE) 2009-06-15T13:45:30 -> Δευτέρα, 15 Ιουνίου 2009 8:45:30 μμ (el-GR)
"Y", "y"	Year month pattern. More information: The Year Month ("Y") Format Specifier.	2009-06-15T13:45:30 -> June, 2009 (en-US) 2009-06-15T13:45:30 -> juni 2009 (da-DK) 2009-06-15T13:45:30 -> Juni 2009 (id-ID)
Any other single character	Unknown specifier.	Throws a run-time FormatException.

How Standard Format Strings Work

In a formatting operation, a standard format string is simply an alias for a custom format string. The advantage of using an alias to refer to a custom format string is that, although the alias remains invariant, the custom format string itself can vary. This is important because the string representations of date and time values typically vary by culture. For example, the "d" standard format string indicates that a date and time value is to be displayed using a short date pattern. For the invariant culture, this pattern is "MM/dd/yyyy". For the fr-FR culture, it is "dd/MM/yyyy". For the ja-JP culture, it is "yyyy/MM/dd".

If a standard format string in a formatting operation maps to a particular culture's custom format string, your application can define the specific culture whose custom format strings are used in one of these ways:

- You can use the default (or current) culture. The following example displays a date using the current culture's short date format. In this case, the current culture is en-US.

  VB

  ```
  ' Display using current (en-us) culture's short date format
  Dim thisDate As Date = #03/15/2008#
  Console.WriteLine(thisDate.ToString("d"))     ' Displays 3/15/2008
  ```

- You can pass a CultureInfo object representing the culture whose formatting is to be used to a method that has an IFormatProvider parameter. The following example displays a date using the short date format of the pt-BR culture.

  VB

  ```
  ' Display using pt-BR culture's short date format
  Dim thisDate As Date = #03/15/2008#
  Dim culture As New CultureInfo("pt-BR")
  Console.WriteLine(thisDate.ToString("d", culture))   ' Displays 15/3/2008
  ```

- You can pass a DateTimeFormatInfo object that provides formatting information to a method that has an IFormatProvider parameter. The following example displays a date using the short date format from a DateTimeFormatInfo object for the hr-HR culture.

  VB

  ```
  ' Display using date format information from hr-HR culture
  Dim thisDate As Date = #03/15/2008#
  Dim fmt As DateTimeFormatInfo = (New CultureInfo("hr-HR")).DateTimeFormat
  Console.WriteLine(thisDate.ToString("d", fmt))   ' Displays 15.3.2008
  ```

Note

For information about customizing the patterns or strings used in formatting date and time values, see the NumberFormatInfo class topic.

In some cases, the standard format string serves as a convenient abbreviation for a longer custom format string that is invariant. Four standard format strings fall into this category: "O" (or "o"), "R" (or "r"), "s", and "u". These strings correspond to custom format strings defined by the invariant culture. They produce string representations of date and time values that are intended to be identical across cultures. The following table provides information on these four standard date and time format strings.

Standard format string	Defined by DateTimeFormatInfo.InvariantInfo property	Custom format string

"O" or "o"	None	yyyy'-'MM'-'dd'T'HH':'mm':'ss'.'fffffffzz
"R" or "r"	RFC1123Pattern	ddd, dd MMM yyyy HH':'mm':'ss 'GMT'
"s"	SortableDateTimePattern	yyyy'-'MM'-'dd'T'HH':'mm':'ss
"u"	UniversalSortableDateTimePattern	yyyy'-'MM'-'dd HH':'mm':'ss'Z'

Standard format strings can also be used in parsing operations with the DateTime.ParseExact or DateTimeOffset.ParseExact methods, which require an input string to exactly conform to a particular pattern for the parse operation to succeed. Many standard format strings map to multiple custom format strings, so a date and time value can be represented in a variety of formats and the parse operation will still succeed. You can determine the custom format string or strings that correspond to a standard format string by calling the DateTimeFormatInfo.GetAllDateTimePatterns(Char) method. The following example displays the custom format strings that map to the "d" (short date pattern) standard format string.

VB

```
Imports System.Globalization

Module Example
   Public Sub Main()
      Console.WriteLine("'d' standard format string:")
      For Each customString In
DateTimeFormatInfo.CurrentInfo.GetAllDateTimePatterns("d"c)
          Console.WriteLine("   {0}", customString)
      Next
   End Sub
End Module
' The example displays the following output:
'    'd' standard format string:
'       M/d/yyyy
'       M/d/yy
'       MM/dd/yy
'       MM/dd/yyyy
'       yy/MM/dd
'       yyyy-MM-dd
'       dd-MMM-yy
```

The following sections describe the standard format specifiers for DateTime and DateTimeOffset values.

The Short Date ("d") Format Specifier

The "d" standard format specifier represents a custom date and time format string that is defined by a specific culture's DateTimeFormatInfo.ShortDatePattern property. For example, the custom format string that is returned by the ShortDatePattern property of the invariant culture is "MM/dd/yyyy".

The following table lists the DateTimeFormatInfo object properties that control the formatting of the returned string.

Property	Description
ShortDatePattern	Defines the overall format of the result string.
DateSeparator	Defines the string that separates the year, month, and day components of a date.

The following example uses the "d" format specifier to display a date and time value.

VB

```
Dim date1 As Date = #4/10/2008#
Console.WriteLine(date1.ToString("d", DateTimeFormatInfo.InvariantInfo))
' Displays 04/10/2008
Console.WriteLine(date1.ToString("d", _
                  CultureInfo.CreateSpecificCulture("en-US")))
' Displays 4/10/2008
Console.WriteLine(date1.ToString("d", _
                  CultureInfo.CreateSpecificCulture("en-NZ")))
' Displays 10/04/2008
Console.WriteLine(date1.ToString("d", _
                  CultureInfo.CreateSpecificCulture("de-DE")))
' Displays 10.04.2008
```

Back to table

The Long Date ("D") Format Specifier

The "D" standard format specifier represents a custom date and time format string that is defined by the current DateTimeFormatInfo.LongDatePattern property. For example, the custom format string for the invariant culture is "dddd, dd MMMM yyyy".

The following table lists the properties of the DateTimeFormatInfo object that control the formatting of the returned string.

Property	Description
LongDatePattern	Defines the overall format of the result string.
DayNames	Defines the localized day names that can appear in the result string.
MonthNames	Defines the localized month names that can appear in the result string.

The following example uses the "D" format specifier to display a date and time value.

VB

```
Dim date1 As Date = #4/10/2008#
Console.WriteLine(date1.ToString("D", _
                  CultureInfo.CreateSpecificCulture("en-US")))
' Displays Thursday, April 10, 2008
Console.WriteLine(date1.ToString("D", _
                  CultureInfo.CreateSpecificCulture("pt-BR")))
' Displays quinta-feira, 10 de abril de 2008
Console.WriteLine(date1.ToString("D", _
                  CultureInfo.CreateSpecificCulture("es-MX")))
' Displays jueves, 10 de abril de 2008
```

Back to table

The Full Date Short Time ("f") Format Specifier

The "f" standard format specifier represents a combination of the long date ("D") and short time ("t") patterns, separated by a space.

The result string is affected by the formatting information of a specific DateTimeFormatInfo object. The following table lists the DateTimeFormatInfo object properties that may control the formatting of the returned string. The custom format specifier returned by the DateTimeFormatInfo.LongDatePattern and DateTimeFormatInfo.ShortTimePattern properties of some cultures may not make use of all properties.

Property	Description
LongDatePattern	Defines the format of the date component of the result string.
ShortTimePattern	Defines the format of the time component of the result string.
DayNames	Defines the localized day names that can appear in the result string.
MonthNames	Defines the localized month names that can appear in the result string.
TimeSeparator	Defines the string that separates the hour, minute, and second components of a time.
AMDesignator	Defines the string that indicates times from midnight to before noon in a 12-hour clock.
PMDesignator	Defines the string that indicates times from noon to before midnight in a 12-hour clock.

The following example uses the "f" format specifier to display a date and time value.

VB

```
Dim date1 As Date = #4/10/2008 6:30AM#
Console.WriteLine(date1.ToString("f", _
```

```
                    CultureInfo.CreateSpecificCulture("en-US")))
' Displays Thursday, April 10, 2008 6:30 AM
Console.WriteLine(date1.ToString("f", _
                    CultureInfo.CreateSpecificCulture("fr-FR")))
' Displays jeudi 10 avril 2008 06:30
```

Back to table

The Full Date Long Time ("F") Format Specifier

The "F" standard format specifier represents a custom date and time format string that is defined by the current DateTimeFormatInfo.FullDateTimePattern property. For example, the custom format string for the invariant culture is "dddd, dd MMMM yyyy HH:mm:ss".

The following table lists the DateTimeFormatInfo object properties that may control the formatting of the returned string. The custom format specifier that is returned by the FullDateTimePattern property of some cultures may not make use of all properties.

Property	Description
FullDateTimePattern	Defines the overall format of the result string.
DayNames	Defines the localized day names that can appear in the result string.
MonthNames	Defines the localized month names that can appear in the result string.
TimeSeparator	Defines the string that separates the hour, minute, and second components of a time.
AMDesignator	Defines the string that indicates times from midnight to before noon in a 12-hour clock.
PMDesignator	Defines the string that indicates times from noon to before midnight in a 12-hour clock.

The following example uses the "F" format specifier to display a date and time value.

VB

```
Dim date1 As Date = #4/10/2008 6:30AM#
Console.WriteLine(date1.ToString("F", _
                    CultureInfo.CreateSpecificCulture("en-US")))
' Displays Thursday, April 10, 2008 6:30:00 AM
Console.WriteLine(date1.ToString("F", _
                    CultureInfo.CreateSpecificCulture("fr-FR")))
' Displays jeudi 10 avril 2008 06:30:00
```

Back to table

The General Date Short Time ("g") Format Specifier

The "g" standard format specifier represents a combination of the short date ("d") and short time ("t") patterns, separated by a space.

The result string is affected by the formatting information of a specific DateTimeFormatInfo object. The following table lists the DateTimeFormatInfo object properties that may control the formatting of the returned string. The custom format specifier that is returned by the DateTimeFormatInfo.ShortDatePattern and DateTimeFormatInfo.ShortTimePattern properties of some cultures may not make use of all properties.

Property	Description
ShortDatePattern	Defines the format of the date component of the result string.
ShortTimePattern	Defines the format of the time component of the result string.
DateSeparator	Defines the string that separates the year, month, and day components of a date.
TimeSeparator	Defines the string that separates the hour, minute, and second components of a time.
AMDesignator	Defines the string that indicates times from midnight to before noon in a 12-hour clock.
PMDesignator	Defines the string that indicates times from noon to before midnight in a 12-hour clock.

The following example uses the "g" format specifier to display a date and time value.

VB

```
Dim date1 As Date = #4/10/2008 6:30AM#
Console.WriteLine(date1.ToString("g", _
                  DateTimeFormatInfo.InvariantInfo))
' Displays 04/10/2008 06:30
Console.WriteLine(date1.ToString("g", _
                  CultureInfo.CreateSpecificCulture("en-us")))
' Displays 4/10/2008 6:30 AM
Console.WriteLine(date1.ToString("g", _
                  CultureInfo.CreateSpecificCulture("fr-BE")))
' Displays 10/04/2008 6:30
```

Back to table

The General Date Long Time ("G") Format Specifier

The "G" standard format specifier represents a combination of the short date ("d") and long time ("T") patterns, separated by a space.

The result string is affected by the formatting information of a specific DateTimeFormatInfo object. The following table

lists the DateTimeFormatInfo object properties that may control the formatting of the returned string. The custom format specifier that is returned by the DateTimeFormatInfo.ShortDatePattern and DateTimeFormatInfo.LongTimePattern properties of some cultures may not make use of all properties.

Property	Description
ShortDatePattern	Defines the format of the date component of the result string.
LongTimePattern	Defines the format of the time component of the result string.
DateSeparator	Defines the string that separates the year, month, and day components of a date.
TimeSeparator	Defines the string that separates the hour, minute, and second components of a time.
AMDesignator	Defines the string that indicates times from midnight to before noon in a 12-hour clock.
PMDesignator	Defines the string that indicates times from noon to before midnight in a 12-hour clock.

The following example uses the "G" format specifier to display a date and time value.

VB

```
Dim date1 As Date = #4/10/2008 6:30AM#
Console.WriteLine(date1.ToString("G", _
                  DateTimeFormatInfo.InvariantInfo))
' Displays 04/10/2008 06:30:00
Console.WriteLine(date1.ToString("G", _
                  CultureInfo.CreateSpecificCulture("en-us")))
' Displays 4/10/2008 6:30:00 AM
Console.WriteLine(date1.ToString("G", _
                  CultureInfo.CreateSpecificCulture("nl-BE")))
' Displays 10/04/2008 6:30:00
```

Back to table

The Month ("M", "m") Format Specifier

The "M" or "m" standard format specifier represents a custom date and time format string that is defined by the current DateTimeFormatInfo.MonthDayPattern property. For example, the custom format string for the invariant culture is "MMMM dd".

The following table lists the DateTimeFormatInfo object properties that control the formatting of the returned string.

Property	Description

MonthDayPattern	Defines the overall format of the result string.
MonthNames	Defines the localized month names that can appear in the result string.

The following example uses the "m" format specifier to display a date and time value.

VB

```
Dim date1 As Date = #4/10/2008 6:30AM#
Console.WriteLine(date1.ToString("m", _
                  CultureInfo.CreateSpecificCulture("en-us")))
' Displays April 10
Console.WriteLine(date1.ToString("m", _
                  CultureInfo.CreateSpecificCulture("ms-MY")))
' Displays 10 April
```

Back to table

The Round-trip ("O", "o") Format Specifier

The "O" or "o" standard format specifier represents a custom date and time format string using a pattern that preserves time zone information and emits a result string that complies with ISO 8601. For DateTime values, this format specifier is designed to preserve date and time values along with the DateTime.Kind property in text. The formatted string can be parsed back by using the DateTime.Parse(String, IFormatProvider, DateTimeStyles) or DateTime.ParseExact method if the *styles* parameter is set to DateTimeStyles.RoundtripKind.

The "O" or "o" standard format specifier corresponds to the "yyyy'-'MM'-'dd'T'HH':'mm':'ss'.'fffffffK" custom format string for DateTime values and to the "yyyy'-'MM'-'dd'T'HH':'mm':'ss'.'fffffffzzz" custom format string for DateTimeOffset values. In this string, the pairs of single quotation marks that delimit individual characters, such as the hyphens, the colons, and the letter "T", indicate that the individual character is a literal that cannot be changed. The apostrophes do not appear in the output string.

The O" or "o" standard format specifier (and the "yyyy'-'MM'-'dd'T'HH':'mm':'ss'.'fffffffK" custom format string) takes advantage of the three ways that ISO 8601 represents time zone information to preserve the Kind property of DateTime values:

- The time zone component of DateTimeKind.Local date and time values is an offset from UTC (for example, +01:00, -07:00). All DateTimeOffset values are also represented in this format.
- The time zone component of DateTimeKind.Utc date and time values uses "Z" (which stands for zero offset) to represent UTC.
- DateTimeKind.Unspecified date and time values have no time zone information.

Because the O" or "o" standard format specifier conforms to an international standard, the formatting or parsing operation that uses the specifier always uses the invariant culture and the Gregorian calendar.

Strings that are passed to the **Parse**, **TryParse**, **ParseExact**, and **TryParseExact** methods of DateTime and

DateTimeOffset can be parsed by using the "O" or "o" format specifier if they are in one of these formats. In the case of DateTime objects, the parsing overload that you call should also include a *styles* parameter with a value of DateTimeStyles.RoundtripKind. Note that if you call a parsing method with the custom format string that corresponds to the "O" or "o" format specifier, you won't get the same results as "O" or "o". This is because parsing methods that use a custom format string can't parse the string representation of date and time values that lack a time zone component or use "Z" to indicate UTC.

The following example uses the "o" format specifier to display a series of DateTime values and a DateTimeOffset value on a system in the U.S. Pacific Time zone.

VB

```
Module Example
   Public Sub Main()
       Dim dat As New Date(2009, 6, 15, 13, 45, 30,
                           DateTimeKind.Unspecified)
       Console.WriteLine("{0} ({1}) --> {0:O}", dat, dat.Kind)

       Dim uDat As New Date(2009, 6, 15, 13, 45, 30, DateTimeKind.Utc)
       Console.WriteLine("{0} ({1}) --> {0:O}", uDat, uDat.Kind)

       Dim lDat As New Date(2009, 6, 15, 13, 45, 30, DateTimeKind.Local)
       Console.WriteLine("{0} ({1}) --> {0:O}", lDat, lDat.Kind)
       Console.WriteLine()

       Dim dto As New DateTimeOffset(lDat)
       Console.WriteLine("{0} --> {0:O}", dto)
   End Sub
End Module
' The example displays the following output:
'    6/15/2009 1:45:30 PM (Unspecified) --> 2009-06-15T13:45:30.0000000
'    6/15/2009 1:45:30 PM (Utc) --> 2009-06-15T13:45:30.0000000Z
'    6/15/2009 1:45:30 PM (Local) --> 2009-06-15T13:45:30.0000000-07:00
'
'    6/15/2009 1:45:30 PM -07:00 --> 2009-06-15T13:45:30.0000000-07:00
```

The following example uses the "o" format specifier to create a formatted string, and then restores the original date and time value by calling a date and time **Parse** method.

VB

```
' Round-trip DateTime values.
Dim originalDate, newDate As Date
Dim dateString As String
' Round-trip a local time.
originalDate = Date.SpecifyKind(#4/10/2008 6:30AM#, DateTimeKind.Local)
dateString = originalDate.ToString("o")
newDate = Date.Parse(dateString, Nothing, DateTimeStyles.RoundtripKind)
Console.WriteLine("Round-tripped {0} {1} to {2} {3}.", originalDate, originalDate.Kind,
_
                  newDate, newDate.Kind)
' Round-trip a UTC time.
originalDate = Date.SpecifyKind(#4/12/2008 9:30AM#, DateTimeKind.Utc)
```

```
dateString = originalDate.ToString("o")
newDate = Date.Parse(dateString, Nothing, DateTimeStyles.RoundtripKind)
Console.WriteLine("Round-tripped {0} {1} to {2} {3}.", originalDate, originalDate.Kind,
_
                  newDate, newDate.Kind)
' Round-trip time in an unspecified time zone.
originalDate = Date.SpecifyKind(#4/13/2008 12:30PM#,
DateTimeKind.Unspecified)
dateString = originalDate.ToString("o")
newDate = Date.Parse(dateString, Nothing, DateTimeStyles.RoundtripKind)
Console.WriteLine("Round-tripped {0} {1} to {2} {3}.", originalDate, originalDate.Kind,
_
                  newDate, newDate.Kind)

' Round-trip a DateTimeOffset value.
Dim originalDTO As New DateTimeOffset(#4/12/2008 9:30AM#, New TimeSpan(-8, 0, 0))
dateString = originalDTO.ToString("o")
Dim newDTO As DateTimeOffset = DateTimeOffset.Parse(dateString, Nothing,
DateTimeStyles.RoundtripKind)
Console.WriteLine("Round-tripped {0} to {1}.", originalDTO, newDTO)
' The example displays the following output:
'    Round-tripped 4/10/2008 6:30:00 AM Local to 4/10/2008 6:30:00 AM Local.
'    Round-tripped 4/12/2008 9:30:00 AM Utc to 4/12/2008 9:30:00 AM Utc.
'    Round-tripped 4/13/2008 12:30:00 PM Unspecified to 4/13/2008 12:30:00 PM
Unspecified.
'    Round-tripped 4/12/2008 9:30:00 AM -08:00 to 4/12/2008 9:30:00 AM -08:00.
```

Back to table

The RFC1123 ("R", "r") Format Specifier

The "R" or "r" standard format specifier represents a custom date and time format string that is defined by the DateTimeFormatInfo.RFC1123Pattern property. The pattern reflects a defined standard, and the property is read-only. Therefore, it is always the same, regardless of the culture used or the format provider supplied. The custom format string is "ddd, dd MMM yyyy HH':'mm':'ss 'GMT'". When this standard format specifier is used, the formatting or parsing operation always uses the invariant culture.

The result string is affected by the following properties of the DateTimeFormatInfo object returned by the DateTimeFormatInfo.InvariantInfo property that represents the invariant culture.

Property	Description
RFC1123Pattern	Defines the format of the result string.
AbbreviatedDayNames	Defines the abbreviated day names that can appear in the result string.
AbbreviatedMonthNames	Defines the abbreviated month names that can appear in the result string.

Although the RFC 1123 standard expresses a time as Coordinated Universal Time (UTC), the formatting operation does not modify the value of the DateTime object that is being formatted. Therefore, you must convert the DateTime value to UTC by calling the DateTime.ToUniversalTime method before you perform the formatting operation. In contrast, DateTimeOffset values perform this conversion automatically; there is no need to call the DateTimeOffset.ToUniversalTime method before the formatting operation.

The following example uses the "r" format specifier to display a DateTime and a DateTimeOffset value on a system in the U.S. Pacific Time zone.

VB

```
Dim date1 As Date = #4/10/2008 6:30AM#
Dim dateOffset As New DateTimeOffset(date1, TimeZoneInfo.Local.GetUtcOffset(date1))
Console.WriteLine(date1.ToUniversalTime.ToString("r"))
' Displays Thu, 10 Apr 2008 13:30:00 GMT
Console.WriteLine(dateOffset.ToUniversalTime.ToString("r"))
' Displays Thu, 10 Apr 2008 13:30:00 GMT
```

Back to table

The Sortable ("s") Format Specifier

The "s" standard format specifier represents a custom date and time format string that is defined by the DateTimeFormatInfo.SortableDateTimePattern property. The pattern reflects a defined standard (ISO 8601), and the property is read-only. Therefore, it is always the same, regardless of the culture used or the format provider supplied. The custom format string is "yyyy'-'MM'-'dd'T'HH':'mm':'ss".

The purpose of the "s" format specifier is to produce result strings that sort consistently in ascending or descending order based on date and time values. As a result, although the "s" standard format specifier represents a date and time value in a consistent format, the formatting operation does not modify the value of the date and time object that is being formatted to reflect its DateTime.Kind property or its DateTimeOffset.Offset value. For example, the result strings produced by formatting the date and time values 2014-11-15T18:32:17+00:00 and 2014-11-15T18:32:17+08:00 are identical.

When this standard format specifier is used, the formatting or parsing operation always uses the invariant culture.

The following example uses the "s" format specifier to display a DateTime and a DateTimeOffset value on a system in the U.S. Pacific Time zone.

VB

```
Dim date1 As Date = #4/10/2008 6:30AM#
Console.WriteLine(date1.ToString("s"))
' Displays 2008-04-10T06:30:00
```

Back to table

The Short Time ("t") Format Specifier

The "t" standard format specifier represents a custom date and time format string that is defined by the current DateTimeFormatInfo.ShortTimePattern property. For example, the custom format string for the invariant culture is "HH:mm".

The result string is affected by the formatting information of a specific DateTimeFormatInfo object. The following table lists the DateTimeFormatInfo object properties that may control the formatting of the returned string. The custom format specifier that is returned by the DateTimeFormatInfo.ShortTimePattern property of some cultures may not make use of all properties.

Property	Description
ShortTimePattern	Defines the format of the time component of the result string.
TimeSeparator	Defines the string that separates the hour, minute, and second components of a time.
AMDesignator	Defines the string that indicates times from midnight to before noon in a 12-hour clock.
PMDesignator	Defines the string that indicates times from noon to before midnight in a 12-hour clock.

The following example uses the "t" format specifier to display a date and time value.

VB

```
Dim date1 As Date = #4/10/2008 6:30AM#
Console.WriteLine(date1.ToString("t", _
                  CultureInfo.CreateSpecificCulture("en-us")))
' Displays 6:30 AM
Console.WriteLine(date1.ToString("t", _
                  CultureInfo.CreateSpecificCulture("es-ES")))
' Displays 6:30
```

Back to table

The Long Time ("T") Format Specifier

The "T" standard format specifier represents a custom date and time format string that is defined by a specific culture's DateTimeFormatInfo.LongTimePattern property. For example, the custom format string for the invariant culture is "HH:mm:ss".

The following table lists the DateTimeFormatInfo object properties that may control the formatting of the returned string. The custom format specifier that is returned by the DateTimeFormatInfo.LongTimePattern property of some cultures may not make use of all properties.

Property	Description
LongTimePattern	Defines the format of the time component of the result string.

TimeSeparator	Defines the string that separates the hour, minute, and second components of a time.
AMDesignator	Defines the string that indicates times from midnight to before noon in a 12-hour clock.
PMDesignator	Defines the string that indicates times from noon to before midnight in a 12-hour clock.

The following example uses the "T" format specifier to display a date and time value.

VB

```
Dim date1 As Date = #4/10/2008 6:30AM#
Console.WriteLine(date1.ToString("T", _
                  CultureInfo.CreateSpecificCulture("en-us")))
' Displays 6:30:00 AM
Console.WriteLine(date1.ToString("T", _
                  CultureInfo.CreateSpecificCulture("es-ES")))
' Displays 6:30:00
```

Back to table

The Universal Sortable ("u") Format Specifier

The "u" standard format specifier represents a custom date and time format string that is defined by the DateTimeFormatInfo.UniversalSortableDateTimePattern property. The pattern reflects a defined standard, and the property is read-only. Therefore, it is always the same, regardless of the culture used or the format provider supplied. The custom format string is "yyyy'-'MM'-'dd HH':'mm':'ss'Z'". When this standard format specifier is used, the formatting or parsing operation always uses the invariant culture.

Although the result string should express a time as Coordinated Universal Time (UTC), no conversion of the original DateTime value is performed during the formatting operation. Therefore, you must convert a DateTime value to UTC by calling the DateTime.ToUniversalTime method before formatting it. In contrast, DateTimeOffset values perform this conversion automatically; there is no need to call the DateTimeOffset.ToUniversalTime method before the formatting operation.

The following example uses the "u" format specifier to display a date and time value.

VB

```
Dim date1 As Date = #4/10/2008 6:30AM#
Console.WriteLine(date1.ToUniversalTime.ToString("u"))
' Displays 2008-04-10 13:30:00Z
```

Back to table

The Universal Full ("U") Format Specifier

The "U" standard format specifier represents a custom date and time format string that is defined by a specified culture's

DateTimeFormatInfo.FullDateTimePattern property. The pattern is the same as the "F" pattern. However, the DateTime value is automatically converted to UTC before it is formatted.

The following table lists the DateTimeFormatInfo object properties that may control the formatting of the returned string. The custom format specifier that is returned by the FullDateTimePattern property of some cultures may not make use of all properties.

Property	Description
FullDateTimePattern	Defines the overall format of the result string.
DayNames	Defines the localized day names that can appear in the result string.
MonthNames	Defines the localized month names that can appear in the result string.
TimeSeparator	Defines the string that separates the hour, minute, and second components of a time.
AMDesignator	Defines the string that indicates times from midnight to before noon in a 12-hour clock.
PMDesignator	Defines the string that indicates times from noon to before midnight in a 12-hour clock.

The "U" format specifier is not supported by the DateTimeOffset type and throws a FormatException if it is used to format a DateTimeOffset value.

The following example uses the "U" format specifier to display a date and time value.

VB

```
Dim date1 As Date = #4/10/2008 6:30AM#
Console.WriteLine(date1.ToString("U", CultureInfo.CreateSpecificCulture("en-US")))
' Displays Thursday, April 10, 2008 1:30:00 PM
Console.WriteLine(date1.ToString("U", CultureInfo.CreateSpecificCulture("sv-FI")))
' Displays den 10 april 2008 13:30:00
```

Back to table

The Year Month ("Y", "y") Format Specifier

The "Y" or "y" standard format specifier represents a custom date and time format string that is defined by the DateTimeFormatInfo.YearMonthPattern property of a specified culture. For example, the custom format string for the invariant culture is "yyyy MMMM".

The following table lists the DateTimeFormatInfo object properties that control the formatting of the returned string.

Property	Description

YearMonthPattern	Defines the overall format of the result string.
MonthNames	Defines the localized month names that can appear in the result string.

The following example uses the "y" format specifier to display a date and time value.

VB

```
Dim date1 As Date = #4/10/2008 6:30AM#
Console.WriteLine(date1.ToString("Y", CultureInfo.CreateSpecificCulture("en-US")))
' Displays April, 2008
Console.WriteLine(date1.ToString("y", CultureInfo.CreateSpecificCulture("af-ZA")))
' Displays April 2008
```

Back to table

Notes

Control Panel Settings

The settings in the **Regional and Language Options** item in Control Panel influence the result string produced by a formatting operation. These settings are used to initialize the DateTimeFormatInfo object associated with the current thread culture, which provides values used to govern formatting. Computers that use different settings generate different result strings.

In addition, if you use the CultureInfo.CultureInfo(String) constructor to instantiate a new CultureInfo object that represents the same culture as the current system culture, any customizations established by the **Regional and Language Options** item in Control Panel will be applied to the new CultureInfo object. You can use the CultureInfo.CultureInfo(String, Boolean) constructor to create a CultureInfo object that does not reflect a system's customizations.

DateTimeFormatInfo Properties

Formatting is influenced by properties of the current DateTimeFormatInfo object, which is provided implicitly by the current thread culture or explicitly by the IFormatProvider parameter of the method that invokes formatting. For the IFormatProvider parameter, your application should specify a CultureInfo object, which represents a culture, or a DateTimeFormatInfo object, which represents a particular culture's date and time formatting conventions. Many of the standard date and time format specifiers are aliases for formatting patterns defined by properties of the current DateTimeFormatInfo object. Your application can change the result produced by some standard date and time format specifiers by changing the corresponding date and time format patterns of the corresponding DateTimeFormatInfo property.

See Also

System.DateTime
System.DateTimeOffset
Formatting Types in the .NET Framework
Custom Date and Time Format Strings
Sample: .NET Framework 4 Formatting Utility

Custom Date and Time Format Strings

.NET Framework (current version)

A date and time format string defines the text representation of a DateTime or DateTimeOffset value that results from a formatting operation . It can also define the representation of a date and time value that is required in a parsing operation in order to successfully convert the string to a date and time. A custom format string consists of one or more custom date and time format specifiers. Any string that is not a standard date and time format string is interpreted as a custom date and time format string.

Custom date and time format strings can be used with both DateTime and DateTimeOffset values.

Tip

You can download the Formatting Utility, an application that enables you to apply format strings to either date and time or numeric values and displays the result string.

In formatting operations, custom date and time format strings can be used either with the **ToString** method of a date and time instance or with a method that supports composite formatting. The following example illustrates both uses.

VB

```
Dim thisDate1 As Date = #6/10/2011#
Console.WriteLine("Today is " + thisDate1.ToString("MMMM dd, yyyy") + ".")

Dim thisDate2 As New DateTimeOffset(2011, 6, 10, 15, 24, 16, TimeSpan.Zero)
Console.WriteLine("The current date and time: {0:MM/dd/yy H:mm:ss zzz}",
                  thisDate2)
' The example displays the following output:
'    Today is June 10, 2011.
'    The current date and time: 06/10/11 15:24:16 +00:00
```

In parsing operations, custom date and time format strings can be used with the DateTime.ParseExact, DateTime.TryParseExact, DateTimeOffset.ParseExact, and DateTimeOffset.TryParseExact methods. These methods require that an input string conform exactly to a particular pattern for the parse operation to succeed. The following example illustrates a call to the DateTimeOffset.ParseExact(String, String, IFormatProvider) method to parse a date that must include a day, a month, and a two-digit year.

VB

```
Imports System.Globalization

Module Example
   Public Sub Main()
      Dim dateValues() As String = { "30-12-2011", "12-30-2011",
                                     "30-12-11", "12-30-11" }
```

```
        Dim pattern As String = "MM-dd-yy"
        Dim parsedDate As Date

        For Each dateValue As String In dateValues
           If DateTime.TryParseExact(dateValue, pattern, Nothing,
                                     DateTimeStyles.None, parsedDate) Then
              Console.WriteLine("Converted '{0}' to {1:d}.",
                                dateValue, parsedDate)
           Else
              Console.WriteLine("Unable to convert '{0}' to a date and time.",
                                dateValue)
           End If
        Next
     End Sub
  End Module
  ' The example displays the following output:
  '    Unable to convert '30-12-2011' to a date and time.
  '    Unable to convert '12-30-2011' to a date and time.
  '    Unable to convert '30-12-11' to a date and time.
  '    Converted '12-30-11' to 12/30/2011.
```

The following table describes the custom date and time format specifiers and displays a result string produced by each format specifier. By default, result strings reflect the formatting conventions of the en-US culture. If a particular format specifier produces a localized result string, the example also notes the culture to which the result string applies. See the Notes section for additional information about using custom date and time format strings.

Format specifier	**Description**	**Examples**
"d"	The day of the month, from 1 through 31. More information: The "d" Custom Format Specifier.	2009-06-01T13:45:30 -> 1 2009-06-15T13:45:30 -> 15
"dd"	The day of the month, from 01 through 31. More information: The "dd" Custom Format Specifier.	2009-06-01T13:45:30 -> 01 2009-06-15T13:45:30 -> 15
"ddd"	The abbreviated name of the day of the week. More information: The "ddd" Custom Format Specifier.	2009-06-15T13:45:30 -> Mon (en-US) 2009-06-15T13:45:30 -> Пн (ru-RU) 2009-06-15T13:45:30 -> lun. (fr-FR)
"dddd"	The full name of the day of the week. More information: The "dddd" Custom Format Specifier.	2009-06-15T13:45:30 -> Monday (en-US) 2009-06-15T13:45:30 -> понедельник (ru-RU) 2009-06-15T13:45:30 -> lundi (fr-FR)

"f"	The tenths of a second in a date and time value. More information: The "f" Custom Format Specifier.	2009-06-15T13:45:30.6170000 -> 6 2009-06-15T13:45:30.05 -> 0
"ff"	The hundredths of a second in a date and time value. More information: The "ff" Custom Format Specifier.	2009-06-15T13:45:30.6170000 -> 61 2009-06-15T13:45:30.0050000 -> 00
"fff"	The milliseconds in a date and time value. More information: The "fff" Custom Format Specifier.	6/15/2009 13:45:30.617 -> 617 6/15/2009 13:45:30.0005 -> 000
"ffff"	The ten thousandths of a second in a date and time value. More information: The "ffff" Custom Format Specifier.	2009-06-15T13:45:30.6175000 -> 6175 2009-06-15T13:45:30.0000500 -> 0000
"fffff"	The hundred thousandths of a second in a date and time value. More information: The "fffff" Custom Format Specifier.	2009-06-15T13:45:30.6175400 -> 61754 6/15/2009 13:45:30.000005 -> 00000
"ffffff"	The millionths of a second in a date and time value. More information: The "ffffff" Custom Format Specifier.	2009-06-15T13:45:30.6175420 -> 617542 2009-06-15T13:45:30.0000005 -> 000000
"fffffff"	The ten millionths of a second in a date and time value. More information: The "fffffff" Custom Format Specifier.	2009-06-15T13:45:30.6175425 -> 6175425 2009-06-15T13:45:30.0001150 -> 0001150
"F"	If non-zero, the tenths of a second in a date and time value. More information: The "F" Custom Format Specifier.	2009-06-15T13:45:30.6170000 -> 6 2009-06-15T13:45:30.0500000 -> (no output)
"FF"	If non-zero, the hundredths of a second in a date and time value. More information: The "FF" Custom Format Specifier.	2009-06-15T13:45:30.6170000 -> 61 2009-06-15T13:45:30.0050000 -> (no output)

"FFF"	If non-zero, the milliseconds in a date and time value. More information: The "FFF" Custom Format Specifier.	2009-06-15T13:45:30.6170000 -> 617 2009-06-15T13:45:30.0005000 -> (no output)
"FFFF"	If non-zero, the ten thousandths of a second in a date and time value. More information: The "FFFF" Custom Format Specifier.	2009-06-15T13:45:30.5275000 -> 5275 2009-06-15T13:45:30.0000500 -> (no output)
"FFFFF"	If non-zero, the hundred thousandths of a second in a date and time value. More information: The "FFFFF" Custom Format Specifier.	2009-06-15T13:45:30.6175400 -> 61754 2009-06-15T13:45:30.0000050 -> (no output)
"FFFFFF"	If non-zero, the millionths of a second in a date and time value. More information: The "FFFFFF" Custom Format Specifier.	2009-06-15T13:45:30.6175420 -> 617542 2009-06-15T13:45:30.0000005 -> (no output)
"FFFFFFF"	If non-zero, the ten millionths of a second in a date and time value. More information: The "FFFFFFF" Custom Format Specifier.	2009-06-15T13:45:30.6175425 -> 6175425 2009-06-15T13:45:30.0001150 -> 000115
"g", "gg"	The period or era. More information: The "g" or "gg" Custom Format Specifier.	2009-06-15T13:45:30.6170000 -> A.D.
"h"	The hour, using a 12-hour clock from 1 to 12. More information: The "h" Custom Format Specifier.	2009-06-15T01:45:30 -> 1 2009-06-15T13:45:30 -> 1
"hh"	The hour, using a 12-hour clock from 01 to 12. More information: The "hh" Custom Format Specifier.	2009-06-15T01:45:30 -> 01 2009-06-15T13:45:30 -> 01
"H"	The hour, using a 24-hour clock from 0 to 23. More information: The "H" Custom Format Specifier.	2009-06-15T01:45:30 -> 1 2009-06-15T13:45:30 -> 13

"HH"	The hour, using a 24-hour clock from 00 to 23. More information: The "HH" Custom Format Specifier.	2009-06-15T01:45:30 -> 01 2009-06-15T13:45:30 -> 13
"K"	Time zone information. More information: The "K" Custom Format Specifier.	With DateTime values: 2009-06-15T13:45:30, Kind Unspecified -> 2009-06-15T13:45:30, Kind Utc -> Z 2009-06-15T13:45:30, Kind Local -> -07:00 (depends on local computer settings) With DateTimeOffset values: 2009-06-15T01:45:30-07:00 --> -07:00 2009-06-15T08:45:30+00:00 --> +00:00
"m"	The minute, from 0 through 59. More information: The "m" Custom Format Specifier.	2009-06-15T01:09:30 -> 9 2009-06-15T13:29:30 -> 29
"mm"	The minute, from 00 through 59. More information: The "mm" Custom Format Specifier.	2009-06-15T01:09:30 -> 09 2009-06-15T01:45:30 -> 45
"M"	The month, from 1 through 12. More information: The "M" Custom Format Specifier.	2009-06-15T13:45:30 -> 6
"MM"	The month, from 01 through 12. More information: The "MM" Custom Format Specifier.	2009-06-15T13:45:30 -> 06
"MMM"	The abbreviated name of the month. More information: The "MMM" Custom Format Specifier.	2009-06-15T13:45:30 -> Jun (en-US) 2009-06-15T13:45:30 -> juin (fr-FR) 2009-06-15T13:45:30 -> Jun (zu-ZA)
"MMMM"	The full name of the month. More information: The "MMMM" Custom Format Specifier.	2009-06-15T13:45:30 -> June (en-US) 2009-06-15T13:45:30 -> juni (da-DK) 2009-06-15T13:45:30 -> uJuni (zu-ZA)

"s"	The second, from 0 through 59. More information: The "s" Custom Format Specifier.	2009-06-15T13:45:09 -> 9
"ss"	The second, from 00 through 59. More information: The "ss" Custom Format Specifier.	2009-06-15T13:45:09 -> 09
"t"	The first character of the AM/PM designator. More information: The "t" Custom Format Specifier.	2009-06-15T13:45:30 -> P (en-US) 2009-06-15T13:45:30 -> 午 (ja-JP) 2009-06-15T13:45:30 -> (fr-FR)
"tt"	The AM/PM designator. More information: The "tt" Custom Format Specifier.	2009-06-15T13:45:30 -> PM (en-US) 2009-06-15T13:45:30 -> 午後 (ja-JP) 2009-06-15T13:45:30 -> (fr-FR)
"y"	The year, from 0 to 99. More information: The "y" Custom Format Specifier.	0001-01-01T00:00:00 -> 1 0900-01-01T00:00:00 -> 0 1900-01-01T00:00:00 -> 0 2009-06-15T13:45:30 -> 9 2019-06-15T13:45:30 -> 19
"yy"	The year, from 00 to 99. More information: The "yy" Custom Format Specifier.	0001-01-01T00:00:00 -> 01 0900-01-01T00:00:00 -> 00 1900-01-01T00:00:00 -> 00 2019-06-15T13:45:30 -> 19
"yyy"	The year, with a minimum of three digits. More information: The "yyy" Custom Format Specifier.	0001-01-01T00:00:00 -> 001 0900-01-01T00:00:00 -> 900 1900-01-01T00:00:00 -> 1900 2009-06-15T13:45:30 -> 2009
"yyyy"	The year as a four-digit number. More information: The "yyyy" Custom Format Specifier.	0001-01-01T00:00:00 -> 0001 0900-01-01T00:00:00 -> 0900

		1900-01-01T00:00:00 -> 1900 2009-06-15T13:45:30 -> 2009
"yyyyy"	The year as a five-digit number. More information: The "yyyyy" Custom Format Specifier.	0001-01-01T00:00:00 -> 00001 2009-06-15T13:45:30 -> 02009
"z"	Hours offset from UTC, with no leading zeros. More information: The "z" Custom Format Specifier.	2009-06-15T13:45:30-07:00 -> -7
"zz"	Hours offset from UTC, with a leading zero for a single-digit value. More information: The "zz" Custom Format Specifier.	2009-06-15T13:45:30-07:00 -> -07
"zzz"	Hours and minutes offset from UTC. More information: The "zzz" Custom Format Specifier.	2009-06-15T13:45:30-07:00 -> -07:00
":"	The time separator. More information: The ":" Custom Format Specifier.	2009-06-15T13:45:30 -> : (en-US) 2009-06-15T13:45:30 -> . (it-IT) 2009-06-15T13:45:30 -> : (ja-JP)
"/"	The date separator. More Information: The "/" Custom Format Specifier.	2009-06-15T13:45:30 -> / (en-US) 2009-06-15T13:45:30 -> - (ar-DZ) 2009-06-15T13:45:30 -> . (tr-TR)
"*string*" '*string*'	Literal string delimiter. More information: Character literals.	2009-06-15T13:45:30 ("arr:" h:m t) -> arr: 1:45 P 2009-06-15T13:45:30 ('arr:' h:m t) -> arr: 1:45 P
%	Defines the following character as a custom format specifier. More information:Using Single Custom Format Specifiers.	2009-06-15T13:45:30 (%h) -> 1
\	The escape character. More information: Character literals and Using the Escape Character.	2009-06-15T13:45:30 (h \h) -> 1 h

Any other character	The character is copied to the result string unchanged. More information: Character literals.	2009-06-15T01:45:30 (arr hh:mm t) -> arr 01:45 A

The following sections provide additional information about each custom date and time format specifier. Unless otherwise noted, each specifier produces an identical string representation regardless of whether it is used with a DateTime value or a DateTimeOffset value.

The "d" custom format specifier

The "d" custom format specifier represents the day of the month as a number from 1 through 31. A single-digit day is formatted without a leading zero.

If the "d" format specifier is used without other custom format specifiers, it is interpreted as the "d" standard date and time format specifier. For more information about using a single format specifier, see Using Single Custom Format Specifiers later in this topic.

The following example includes the "d" custom format specifier in several format strings.

VB

```
Dim date1 As Date = #08/29/2008 7:27:15PM#

Console.WriteLine(date1.ToString("d, M", _
                  CultureInfo.InvariantCulture))
' Displays 29, 8

Console.WriteLine(date1.ToString("d MMMM", _
                  CultureInfo.CreateSpecificCulture("en-US")))
' Displays 29 August
Console.WriteLine(date1.ToString("d MMMM", _
                  CultureInfo.CreateSpecificCulture("es-MX")))
' Displays 29 agosto
```

Back to table

The "dd" custom format specifier

The "dd" custom format string represents the day of the month as a number from 01 through 31. A single-digit day is formatted with a leading zero.

The following example includes the "dd" custom format specifier in a custom format string.

VB

```
Dim date1 As Date = #1/2/2008 6:30:15AM#

Console.WriteLine(date1.ToString("dd, MM", _
```

```
                    CultureInfo.InvariantCulture))
' 02, 01
```

Back to table

The "ddd" custom format specifier

The "ddd" custom format specifier represents the abbreviated name of the day of the week. The localized abbreviated name of the day of the week is retrieved from the DateTimeFormatInfo.AbbreviatedDayNames property of the current or specified culture.

The following example includes the "ddd" custom format specifier in a custom format string.

VB

```
Dim date1 As Date = #08/29/2008 7:27:15PM#

Console.WriteLine(date1.ToString("ddd d MMM", _
                  CultureInfo.CreateSpecificCulture("en-US")))
' Displays Fri 29 Aug
Console.WriteLine(date1.ToString("ddd d MMM", _
                  CultureInfo.CreateSpecificCulture("fr-FR")))
' Displays ven. 29 août
```

Back to table

The "dddd" custom format specifier

The "dddd" custom format specifier (plus any number of additional "d" specifiers) represents the full name of the day of the week. The localized name of the day of the week is retrieved from the DateTimeFormatInfo.DayNames property of the current or specified culture.

The following example includes the "dddd" custom format specifier in a custom format string.

VB

```
Dim date1 As Date = #08/29/2008 7:27:15PM#

Console.WriteLine(date1.ToString("dddd dd MMMM", _
                  CultureInfo.CreateSpecificCulture("en-US")))
' Displays Friday 29 August
Console.WriteLine(date1.ToString("dddd dd MMMM", _
                  CultureInfo.CreateSpecificCulture("it-IT")))
' Displays venerdì 29 agosto
```

Back to table

The "f" custom format specifier

The "f" custom format specifier represents the most significant digit of the seconds fraction; that is, it represents the tenths of a second in a date and time value.

If the "f" format specifier is used without other format specifiers, it is interpreted as the "f" standard date and time format specifier. For more information about using a single format specifier, see Using Single Custom Format Specifiers later in this topic.

When you use "f" format specifiers as part of a format string supplied to the ParseExact, TryParseExact, ParseExact, or TryParseExact method, the number of "f" format specifiers indicates the number of most significant digits of the seconds fraction that must be present to successfully parse the string.

The following example includes the "f" custom format specifier in a custom format string.

VB

```
Dim date1 As New Date(2008, 8, 29, 19, 27, 15, 018)
Dim ci As CultureInfo = CultureInfo.InvariantCulture

Console.WriteLine(date1.ToString("hh:mm:ss.f", ci))
' Displays 07:27:15.0
Console.WriteLine(date1.ToString("hh:mm:ss.F", ci))
' Displays 07:27:15
Console.WriteLine(date1.ToString("hh:mm:ss.ff", ci))
' Displays 07:27:15.01
Console.WriteLine(date1.ToString("hh:mm:ss.FF", ci))
' Displays 07:27:15.01
Console.WriteLine(date1.ToString("hh:mm:ss.fff", ci))
' Displays 07:27:15.018
Console.WriteLine(date1.ToString("hh:mm:ss.FFF", ci))
' Displays 07:27:15.018
```

Back to table

The "ff" custom format specifier

The "ff" custom format specifier represents the two most significant digits of the seconds fraction; that is, it represents the hundredths of a second in a date and time value.

following example includes the "ff" custom format specifier in a custom format string.

VB

```
Dim date1 As New Date(2008, 8, 29, 19, 27, 15, 018)
Dim ci As CultureInfo = CultureInfo.InvariantCulture

Console.WriteLine(date1.ToString("hh:mm:ss.f", ci))
' Displays 07:27:15.0
Console.WriteLine(date1.ToString("hh:mm:ss.F", ci))
```

```
' Displays 07:27:15
Console.WriteLine(date1.ToString("hh:mm:ss.ff", ci))
' Displays 07:27:15.01
Console.WriteLine(date1.ToString("hh:mm:ss.FF", ci))
' Displays 07:27:15.01
Console.WriteLine(date1.ToString("hh:mm:ss.fff", ci))
' Displays 07:27:15.018
Console.WriteLine(date1.ToString("hh:mm:ss.FFF", ci))
' Displays 07:27:15.018
```

Back to table

The "fff" custom format specifier

The "fff" custom format specifier represents the three most significant digits of the seconds fraction; that is, it represents the milliseconds in a date and time value.

The following example includes the "fff" custom format specifier in a custom format string.

VB

```
Dim date1 As New Date(2008, 8, 29, 19, 27, 15, 018)
Dim ci As CultureInfo = CultureInfo.InvariantCulture

Console.WriteLine(date1.ToString("hh:mm:ss.f", ci))
' Displays 07:27:15.0
Console.WriteLine(date1.ToString("hh:mm:ss.F", ci))
' Displays 07:27:15
Console.WriteLine(date1.ToString("hh:mm:ss.ff", ci))
' Displays 07:27:15.01
Console.WriteLine(date1.ToString("hh:mm:ss.FF", ci))
' Displays 07:27:15.01
Console.WriteLine(date1.ToString("hh:mm:ss.fff", ci))
' Displays 07:27:15.018
Console.WriteLine(date1.ToString("hh:mm:ss.FFF", ci))
' Displays 07:27:15.018
```

Back to table

The "ffff" custom format specifier

The "ffff" custom format specifier represents the four most significant digits of the seconds fraction; that is, it represents the ten thousandths of a second in a date and time value.

Although it is possible to display the ten thousandths of a second component of a time value, that value may not be meaningful. The precision of date and time values depends on the resolution of the system clock. On the Windows NT version 3.5 (and later) and Windows Vista operating systems, the clock's resolution is approximately 10-15 milliseconds.

Back to table

The "fffff" custom format specifier

The "fffff" custom format specifier represents the five most significant digits of the seconds fraction; that is, it represents the hundred thousandths of a second in a date and time value.

Although it is possible to display the hundred thousandths of a second component of a time value, that value may not be meaningful. The precision of date and time values depends on the resolution of the system clock. On the Windows NT 3.5 (and later) and Windows Vista operating systems, the clock's resolution is approximately 10-15 milliseconds.

Back to table

The "ffffff" custom format specifier

The "ffffff" custom format specifier represents the six most significant digits of the seconds fraction; that is, it represents the millionths of a second in a date and time value.

Although it is possible to display the millionths of a second component of a time value, that value may not be meaningful. The precision of date and time values depends on the resolution of the system clock. On the Windows NT 3.5 (and later) and Windows Vista operating systems, the clock's resolution is approximately 10-15 milliseconds.

Back to table

The "fffffff" custom format specifier

The "fffffff" custom format specifier represents the seven most significant digits of the seconds fraction; that is, it represents the ten millionths of a second in a date and time value.

Although it is possible to display the ten millionths of a second component of a time value, that value may not be meaningful. The precision of date and time values depends on the resolution of the system clock. On the Windows NT 3.5 (and later) and Windows Vista operating systems, the clock's resolution is approximately 10-15 milliseconds.

Back to table

The "F" custom format specifier

The "F" custom format specifier represents the most significant digit of the seconds fraction; that is, it represents the tenths of a second in a date and time value. Nothing is displayed if the digit is zero.

If the "F" format specifier is used without other format specifiers, it is interpreted as the "F" standard date and time format specifier. For more information about using a single format specifier, see Using Single Custom Format Specifiers later in this topic.

The number of "F" format specifiers used with the ParseExact, TryParseExact, ParseExact, or TryParseExact method

indicates the maximum number of most significant digits of the seconds fraction that can be present to successfully parse the string.

The following example includes the "F" custom format specifier in a custom format string.

VB

```
Dim date1 As New Date(2008, 8, 29, 19, 27, 15, 018)
Dim ci As CultureInfo = CultureInfo.InvariantCulture

Console.WriteLine(date1.ToString("hh:mm:ss.f", ci))
' Displays 07:27:15.0
Console.WriteLine(date1.ToString("hh:mm:ss.F", ci))
' Displays 07:27:15
Console.WriteLine(date1.ToString("hh:mm:ss.ff", ci))
' Displays 07:27:15.01
Console.WriteLine(date1.ToString("hh:mm:ss.FF", ci))
' Displays 07:27:15.01
Console.WriteLine(date1.ToString("hh:mm:ss.fff", ci))
' Displays 07:27:15.018
Console.WriteLine(date1.ToString("hh:mm:ss.FFF", ci))
' Displays 07:27:15.018
```

Back to table

The "FF" custom format specifier

The "FF" custom format specifier represents the two most significant digits of the seconds fraction; that is, it represents the hundredths of a second in a date and time value. However, trailing zeros or two zero digits are not displayed.

The following example includes the "FF" custom format specifier in a custom format string.

VB

```
Dim date1 As New Date(2008, 8, 29, 19, 27, 15, 018)
Dim ci As CultureInfo = CultureInfo.InvariantCulture

Console.WriteLine(date1.ToString("hh:mm:ss.f", ci))
' Displays 07:27:15.0
Console.WriteLine(date1.ToString("hh:mm:ss.F", ci))
' Displays 07:27:15
Console.WriteLine(date1.ToString("hh:mm:ss.ff", ci))
' Displays 07:27:15.01
Console.WriteLine(date1.ToString("hh:mm:ss.FF", ci))
' Displays 07:27:15.01
Console.WriteLine(date1.ToString("hh:mm:ss.fff", ci))
' Displays 07:27:15.018
Console.WriteLine(date1.ToString("hh:mm:ss.FFF", ci))
' Displays 07:27:15.018
```

Back to table

The "FFF" custom format specifier

The "FFF" custom format specifier represents the three most significant digits of the seconds fraction; that is, it represents the milliseconds in a date and time value. However, trailing zeros or three zero digits are not displayed.

The following example includes the "FFF" custom format specifier in a custom format string.

VB

```
Dim date1 As New Date(2008, 8, 29, 19, 27, 15, 018)
Dim ci As CultureInfo = CultureInfo.InvariantCulture

Console.WriteLine(date1.ToString("hh:mm:ss.f", ci))
' Displays 07:27:15.0
Console.WriteLine(date1.ToString("hh:mm:ss.F", ci))
' Displays 07:27:15
Console.WriteLine(date1.ToString("hh:mm:ss.ff", ci))
' Displays 07:27:15.01
Console.WriteLine(date1.ToString("hh:mm:ss.FF", ci))
' Displays 07:27:15.01
Console.WriteLine(date1.ToString("hh:mm:ss.fff", ci))
' Displays 07:27:15.018
Console.WriteLine(date1.ToString("hh:mm:ss.FFF", ci))
' Displays 07:27:15.018
```

Back to table

The "FFFF" custom format specifier

The "FFFF" custom format specifier represents the four most significant digits of the seconds fraction; that is, it represents the ten thousandths of a second in a date and time value. However, trailing zeros or four zero digits are not displayed.

Although it is possible to display the ten thousandths of a second component of a time value, that value may not be meaningful. The precision of date and time values depends on the resolution of the system clock. On the Windows NT 3.5 (and later) and Windows Vista operating systems, the clock's resolution is approximately 10-15 milliseconds.

Back to table

The "FFFFF" custom format specifier

The "FFFFF" custom format specifier represents the five most significant digits of the seconds fraction; that is, it represents the hundred thousandths of a second in a date and time value. However, trailing zeros or five zero digits are not displayed.

Although it is possible to display the hundred thousandths of a second component of a time value, that value may not be

meaningful. The precision of date and time values depends on the resolution of the system clock. On the Windows NT 3.5 (and later) and Windows Vista operating systems, the clock's resolution is approximately 10-15 milliseconds.

Back to table

The "FFFFFF" custom format specifier

The "FFFFFF" custom format specifier represents the six most significant digits of the seconds fraction; that is, it represents the millionths of a second in a date and time value. However, trailing zeros or six zero digits are not displayed.

Although it is possible to display the millionths of a second component of a time value, that value may not be meaningful. The precision of date and time values depends on the resolution of the system clock. On tfhe Windows NT 3.5 (and later) and Windows Vista operating systems, the clock's resolution is approximately 10-15 milliseconds.

Back to table

The "FFFFFFF" custom format specifier

The "FFFFFFF" custom format specifier represents the seven most significant digits of the seconds fraction; that is, it represents the ten millionths of a second in a date and time value. However, trailing zeros or seven zero digits are not displayed.

Although it is possible to display the ten millionths of a second component of a time value, that value may not be meaningful. The precision of date and time values depends on the resolution of the system clock. On the Windows NT 3.5 (and later) and Windows Vista operating systems, the clock's resolution is approximately 10-15 milliseconds.

Back to table

The "g" or "gg" custom format specifier

The "g" or "gg" custom format specifiers (plus any number of additional "g" specifiers) represents the period or era, such as A.D. The formatting operation ignores this specifier if the date to be formatted does not have an associated period or era string.

If the "g" format specifier is used without other custom format specifiers, it is interpreted as the "g" standard date and time format specifier. For more information about using a single format specifier, see Using Single Custom Format Specifiers later in this topic.

The following example includes the "g" custom format specifier in a custom format string.

VB

```
Dim date1 As Date = #08/04/0070#

Console.WriteLine(date1.ToString("MM/dd/yyyy g", _
                  CultureInfo.InvariantCulture))
' Displays 08/04/0070 A.D.
Console.WriteLine(date1.ToString("MM/dd/yyyy g", _
```

```
                    CultureInfo.CreateSpecificCulture("fr-FR")))
' Displays 08/04/0070 ap. J.-C.
```

Back to table

The "h" custom format specifier

The "h" custom format specifier represents the hour as a number from 1 through 12; that is, the hour is represented by a 12-hour clock that counts the whole hours since midnight or noon. A particular hour after midnight is indistinguishable from the same hour after noon. The hour is not rounded, and a single-digit hour is formatted without a leading zero. For example, given a time of 5:43 in the morning or afternoon, this custom format specifier displays "5".

If the "h" format specifier is used without other custom format specifiers, it is interpreted as a standard date and time format specifier and throws a FormatException. For more information about using a single format specifier, see Using Single Custom Format Specifiers later in this topic.

The following example includes the "h" custom format specifier in a custom format string.

VB

```
Dim date1 As Date
date1 = #6:09:01PM#
Console.WriteLine(date1.ToString("h:m:s.F t", _
                  CultureInfo.InvariantCulture))
' Displays 6:9:1 P
Console.WriteLine(date1.ToString("h:m:s.F t", _
                  CultureInfo.CreateSpecificCulture("el-GR")))
' Displays 6:9:1 μ
date1 = New Date(2008, 1, 1, 18, 9, 1, 500)
Console.WriteLine(date1.ToString("h:m:s.F t", _
                  CultureInfo.InvariantCulture))
' Displays 6:9:1.5 P
Console.WriteLine(date1.ToString("h:m:s.F t", _
                  CultureInfo.CreateSpecificCulture("el-GR")))
' Displays 6:9:1.5 μ
```

Back to table

The "hh" custom format specifier

The "hh" custom format specifier (plus any number of additional "h" specifiers) represents the hour as a number from 01 through 12; that is, the hour is represented by a 12-hour clock that counts the whole hours since midnight or noon. A particular hour after midnight is indistinguishable from the same hour after noon. The hour is not rounded, and a single-digit hour is formatted with a leading zero. For example, given a time of 5:43 in the morning or afternoon, this format specifier displays "05".

The following example includes the "hh" custom format specifier in a custom format string.

VB

```
Dim date1 As Date
date1 = #6:09:01PM#
Console.WriteLine(date1.ToString("hh:mm:ss tt", _
                  CultureInfo.InvariantCulture))
' Displays 06:09:01 PM
Console.WriteLine(date1.ToString("hh:mm:ss tt", _
                  CultureInfo.CreateSpecificCulture("hu-HU")))
' Displays 06:09:01 du.
date1 = New Date(2008, 1, 1, 18, 9, 1, 500)
Console.WriteLine(date1.ToString("hh:mm:ss.ff tt", _
                  CultureInfo.InvariantCulture))
' Displays 06:09:01.50 PM
Console.WriteLine(date1.ToString("hh:mm:ss.ff tt", _
                  CultureInfo.CreateSpecificCulture("hu-HU")))
' Displays 06:09:01.50 du.
```

Back to table

The "H" custom format specifier

The "H" custom format specifier represents the hour as a number from 0 through 23; that is, the hour is represented by a zero-based 24-hour clock that counts the hours since midnight. A single-digit hour is formatted without a leading zero.

If the "H" format specifier is used without other custom format specifiers, it is interpreted as a standard date and time format specifier and throws a FormatException. For more information about using a single format specifier, see Using Single Custom Format Specifiers later in this topic.

The following example includes the "H" custom format specifier in a custom format string.

VB

```
Dim date1 As Date = #6:09:01AM#
Console.WriteLine(date1.ToString("H:mm:ss", _
                  CultureInfo.InvariantCulture))
' Displays 6:09:01
```

Back to table

The "HH" custom format specifier

The "HH" custom format specifier (plus any number of additional "H" specifiers) represents the hour as a number from 00 through 23; that is, the hour is represented by a zero-based 24-hour clock that counts the hours since midnight. A single-digit hour is formatted with a leading zero.

The following example includes the "HH" custom format specifier in a custom format string.

VB

```
Dim date1 As Date = #6:09:01AM#
Console.WriteLine(date1.ToString("HH:mm:ss", _
                  CultureInfo.InvariantCulture))
' Displays 06:09:01
```

Back to table

The "K" custom format specifier

The "K" custom format specifier represents the time zone information of a date and time value. When this format specifier is used with DateTime values, the result string is defined by the value of the DateTime.Kind property:

- For the local time zone (a DateTime.Kind property value of DateTimeKind.Local), this specifier is equivalent to the "zzz" specifier and produces a result string containing the local offset from Coordinated Universal Time (UTC); for example, "-07:00".
- For a UTC time (a DateTime.Kind property value of DateTimeKind.Utc), the result string includes a "Z" character to represent a UTC date.
- For a time from an unspecified time zone (a time whose DateTime.Kind property equals DateTimeKind.Unspecified), the result is equivalent to String.Empty.

For DateTimeOffset values, the "K" format specifier is equivalent to the "zz" format specifier, and produces a result string containing the DateTimeOffset value's offset from UTC.

If the "K" format specifier is used without other custom format specifiers, it is interpreted as a standard date and time format specifier and throws a FormatException. For more information about using a single format specifier, see Using Single Custom Format Specifiers later in this topic.

The following example displays the string that results from using the "K" custom format specifier with various DateTime and DateTimeOffset values on a system in the U.S. Pacific Time zone.

VB

```
Console.WriteLine(Date.Now.ToString("%K"))
' Displays -07:00
Console.WriteLine(Date.UtcNow.ToString("%K"))
' Displays Z
Console.WriteLine("'{0}'", _
                  Date.SpecifyKind(Date.Now, _
                                   DateTimeKind.Unspecified). _
                  ToString("%K"))
' Displays ''
Console.WriteLine(DateTimeOffset.Now.ToString("%K"))
' Displays -07:00
Console.WriteLine(DateTimeOffset.UtcNow.ToString("%K"))
' Displays +00:00
Console.WriteLine(New DateTimeOffset(2008, 5, 1, 6, 30, 0, _
                                     New TimeSpan(5, 0, 0)). _
```

```
                    ToString("%K"))
' Displays +05:00
```

Back to table

The "m" custom format specifier

The "m" custom format specifier represents the minute as a number from 0 through 59. The minute represents whole minutes that have passed since the last hour. A single-digit minute is formatted without a leading zero.

If the "m" format specifier is used without other custom format specifiers, it is interpreted as the "m" standard date and time format specifier. For more information about using a single format specifier, see Using Single Custom Format Specifiers later in this topic.

The following example includes the "m" custom format specifier in a custom format string.

VB

```
Dim date1 As Date
date1 = #6:09:01PM#
Console.WriteLine(date1.ToString("h:m:s.F t", _
                  CultureInfo.InvariantCulture))
' Displays 6:9:1 P
Console.WriteLine(date1.ToString("h:m:s.F t", _
                  CultureInfo.CreateSpecificCulture("el-GR")))
' Displays 6:9:1 μ
date1 = New Date(2008, 1, 1, 18, 9, 1, 500)
Console.WriteLine(date1.ToString("h:m:s.F t", _
                  CultureInfo.InvariantCulture))
' Displays 6:9:1.5 P
Console.WriteLine(date1.ToString("h:m:s.F t", _
                  CultureInfo.CreateSpecificCulture("el-GR")))
' Displays 6:9:1.5 μ
```

Back to table

The "mm" custom format specifier

The "mm" custom format specifier (plus any number of additional "m" specifiers) represents the minute as a number from 00 through 59. The minute represents whole minutes that have passed since the last hour. A single-digit minute is formatted with a leading zero.

The following example includes the "mm" custom format specifier in a custom format string.

VB

```
Dim date1 As Date
date1 = #6:09:01PM#
```

```
Console.WriteLine(date1.ToString("hh:mm:ss tt", _
                  CultureInfo.InvariantCulture))
' Displays 06:09:01 PM
Console.WriteLine(date1.ToString("hh:mm:ss tt", _
                  CultureInfo.CreateSpecificCulture("hu-HU")))
' Displays 06:09:01 du.
date1 = New Date(2008, 1, 1, 18, 9, 1, 500)
Console.WriteLine(date1.ToString("hh:mm:ss.ff tt", _
                  CultureInfo.InvariantCulture))
' Displays 06:09:01.50 PM
Console.WriteLine(date1.ToString("hh:mm:ss.ff tt", _
                  CultureInfo.CreateSpecificCulture("hu-HU")))
' Displays 06:09:01.50 du.
```

Back to table

The "M" custom format specifier

The "M" custom format specifier represents the month as a number from 1 through 12 (or from 1 through 13 for calendars that have 13 months). A single-digit month is formatted without a leading zero.

If the "M" format specifier is used without other custom format specifiers, it is interpreted as the "M" standard date and time format specifier. For more information about using a single format specifier, see Using Single Custom Format Specifiers later in this topic.

The following example includes the "M" custom format specifier in a custom format string.

VB

```
Dim date1 As Date = #8/18/2008#
Console.WriteLine(date1.ToString("(M) MMM, MMMM", _
                  CultureInfo.CreateSpecificCulture("en-US")))
' Displays (8) Aug, August
Console.WriteLine(date1.ToString("(M) MMM, MMMM", _
                  CultureInfo.CreateSpecificCulture("nl-NL")))
' Displays (8) aug, augustus
Console.WriteLine(date1.ToString("(M) MMM, MMMM", _
                  CultureInfo.CreateSpecificCulture("lv-LV")))
' Displays (8) Aug, augusts
```

Back to table

The "MM" custom format specifier

The "MM" custom format specifier represents the month as a number from 01 through 12 (or from 1 through 13 for calendars that have 13 months). A single-digit month is formatted with a leading zero.

The following example includes the "MM" custom format specifier in a custom format string.

VB

```
Dim date1 As Date = #1/2/2008 6:30:15AM#

Console.WriteLine(date1.ToString("dd, MM", _
                  CultureInfo.InvariantCulture))
' 02, 01
```

Back to table

The "MMM" custom format specifier

The "MMM" custom format specifier represents the abbreviated name of the month. The localized abbreviated name of the month is retrieved from the DateTimeFormatInfo.AbbreviatedMonthNames property of the current or specified culture.

The following example includes the "MMM" custom format specifier in a custom format string.

VB

```
Dim date1 As Date = #08/29/2008 7:27:15PM#

Console.WriteLine(date1.ToString("ddd d MMM", _
                  CultureInfo.CreateSpecificCulture("en-US")))
' Displays Fri 29 Aug
Console.WriteLine(date1.ToString("ddd d MMM", _
                  CultureInfo.CreateSpecificCulture("fr-FR")))
' Displays ven. 29 août
```

Back to table

The "MMMM" custom format specifier

The "MMMM" custom format specifier represents the full name of the month. The localized name of the month is retrieved from the DateTimeFormatInfo.MonthNames property of the current or specified culture.

The following example includes the "MMMM" custom format specifier in a custom format string.

VB

```
Dim date1 As Date = #08/29/2008 7:27:15PM#

Console.WriteLine(date1.ToString("dddd dd MMMM", _
                  CultureInfo.CreateSpecificCulture("en-US")))
' Displays Friday 29 August
Console.WriteLine(date1.ToString("dddd dd MMMM", _
                  CultureInfo.CreateSpecificCulture("it-IT")))
' Displays venerdì 29 agosto
```

Back to table

The "s" custom format specifier

The "s" custom format specifier represents the seconds as a number from 0 through 59. The result represents whole seconds that have passed since the last minute. A single-digit second is formatted without a leading zero.

If the "s" format specifier is used without other custom format specifiers, it is interpreted as the "s" standard date and time format specifier. For more information about using a single format specifier, see Using Single Custom Format Specifiers later in this topic.

The following example includes the "s" custom format specifier in a custom format string.

VB

```
Dim date1 As Date
date1 = #6:09:01PM#
Console.WriteLine(date1.ToString("h:m:s.F t", _
                  CultureInfo.InvariantCulture))
' Displays 6:9:1 P
Console.WriteLine(date1.ToString("h:m:s.F t", _
                  CultureInfo.CreateSpecificCulture("el-GR")))
' Displays 6:9:1 μ
date1 = New Date(2008, 1, 1, 18, 9, 1, 500)
Console.WriteLine(date1.ToString("h:m:s.F t", _
                  CultureInfo.InvariantCulture))
' Displays 6:9:1.5 P
Console.WriteLine(date1.ToString("h:m:s.F t", _
                  CultureInfo.CreateSpecificCulture("el-GR")))
' Displays 6:9:1.5 μ
```

Back to table

The "ss" custom format specifier

The "ss" custom format specifier (plus any number of additional "s" specifiers) represents the seconds as a number from 00 through 59. The result represents whole seconds that have passed since the last minute. A single-digit second is formatted with a leading zero.

The following example includes the "ss" custom format specifier in a custom format string.

VB

```
Dim date1 As Date
date1 = #6:09:01PM#
Console.WriteLine(date1.ToString("hh:mm:ss tt", _
                  CultureInfo.InvariantCulture))
' Displays 06:09:01 PM
Console.WriteLine(date1.ToString("hh:mm:ss tt", _
```

```
                    CultureInfo.CreateSpecificCulture("hu-HU")))
' Displays 06:09:01 du.
date1 = New Date(2008, 1, 1, 18, 9, 1, 500)
Console.WriteLine(date1.ToString("hh:mm:ss.ff tt", _
                    CultureInfo.InvariantCulture))
' Displays 06:09:01.50 PM
Console.WriteLine(date1.ToString("hh:mm:ss.ff tt", _
                    CultureInfo.CreateSpecificCulture("hu-HU")))
' Displays 06:09:01.50 du.
```

Back to table

The "t" custom format specifier

The "t" custom format specifier represents the first character of the AM/PM designator. The appropriate localized designator is retrieved from the DateTimeFormatInfo.AMDesignator or DateTimeFormatInfo.PMDesignator property of the current or specific culture. The AM designator is used for all times from 0:00:00 (midnight) to 11:59:59.999. The PM designator is used for all times from 12:00:00 (noon) to 23:59:59.999.

If the "t" format specifier is used without other custom format specifiers, it is interpreted as the "t" standard date and time format specifier. For more information about using a single format specifier, see Using Single Custom Format Specifiers later in this topic.

The following example includes the "t" custom format specifier in a custom format string.

VB

```
Dim date1 As Date
date1 = #6:09:01PM#
Console.WriteLine(date1.ToString("h:m:s.F t", _
                    CultureInfo.InvariantCulture))
' Displays 6:9:1 P
Console.WriteLine(date1.ToString("h:m:s.F t", _
                    CultureInfo.CreateSpecificCulture("el-GR")))
' Displays 6:9:1 μ
date1 = New Date(2008, 1, 1, 18, 9, 1, 500)
Console.WriteLine(date1.ToString("h:m:s.F t", _
                    CultureInfo.InvariantCulture))
' Displays 6:9:1.5 P
Console.WriteLine(date1.ToString("h:m:s.F t", _
                    CultureInfo.CreateSpecificCulture("el-GR")))
' Displays 6:9:1.5 μ
```

Back to table

The "tt" custom format specifier

The "tt" custom format specifier (plus any number of additional "t" specifiers) represents the entire AM/PM designator.

The appropriate localized designator is retrieved from the DateTimeFormatInfo.AMDesignator or DateTimeFormatInfo.PMDesignator property of the current or specific culture. The AM designator is used for all times from 0:00:00 (midnight) to 11:59:59.999. The PM designator is used for all times from 12:00:00 (noon) to 23:59:59.999.

Make sure to use the "tt" specifier for languages for which it is necessary to maintain the distinction between AM and PM. An example is Japanese, for which the AM and PM designators differ in the second character instead of the first character.

The following example includes the "tt" custom format specifier in a custom format string.

VB

```
Dim date1 As Date
date1 = #6:09:01PM#
Console.WriteLine(date1.ToString("hh:mm:ss tt", _
                  CultureInfo.InvariantCulture))
' Displays 06:09:01 PM
Console.WriteLine(date1.ToString("hh:mm:ss tt", _
                  CultureInfo.CreateSpecificCulture("hu-HU")))
' Displays 06:09:01 du.
date1 = New Date(2008, 1, 1, 18, 9, 1, 500)
Console.WriteLine(date1.ToString("hh:mm:ss.ff tt", _
                  CultureInfo.InvariantCulture))
' Displays 06:09:01.50 PM
Console.WriteLine(date1.ToString("hh:mm:ss.ff tt", _
                  CultureInfo.CreateSpecificCulture("hu-HU")))
' Displays 06:09:01.50 du.
```

Back to table

The "y" custom format specifier

The "y" custom format specifier represents the year as a one-digit or two-digit number. If the year has more than two digits, only the two low-order digits appear in the result. If the first digit of a two-digit year begins with a zero (for example, 2008), the number is formatted without a leading zero.

If the "y" format specifier is used without other custom format specifiers, it is interpreted as the "y" standard date and time format specifier. For more information about using a single format specifier, see Using Single Custom Format Specifiers later in this topic.

The following example includes the "y" custom format specifier in a custom format string.

VB

```
Dim date1 As Date = #12/1/0001#
Dim date2 As Date = #1/1/2010#
Console.WriteLine(date1.ToString("%y"))
' Displays 1
Console.WriteLine(date1.ToString("yy"))
' Displays 01
Console.WriteLine(date1.ToString("yyy"))
' Displays 001
```

```
Console.WriteLine(date1.ToString("yyyy"))
' Displays 0001
Console.WriteLine(date1.ToString("yyyyy"))
' Displays 00001
Console.WriteLine(date2.ToString("%y"))
' Displays 10
Console.WriteLine(date2.ToString("yy"))
' Displays 10
Console.WriteLine(date2.ToString("yyy"))
' Displays 2010
Console.WriteLine(date2.ToString("yyyy"))
' Displays 2010
Console.WriteLine(date2.ToString("yyyyy"))
' Displays 02010
```

Back to table

The "yy" custom format specifier

The "yy" custom format specifier represents the year as a two-digit number. If the year has more than two digits, only the two low-order digits appear in the result. If the two-digit year has fewer than two significant digits, the number is padded with leading zeros to produce two digits.

In a parsing operation, a two-digit year that is parsed using the "yy" custom format specifier is interpreted based on the Calendar.TwoDigitYearMax property of the format provider's current calendar. The following example parses the string representation of a date that has a two-digit year by using the default Gregorian calendar of the en-US culture, which, in this case, is the current culture. It then changes the current culture's CultureInfo object to use a GregorianCalendar object whose TwoDigitYearMax property has been modified.

VB

```
Imports System.Globalization
Imports System.Threading

Module Example
   Public Sub Main()
      Dim fmt As String = "dd-MMM-yy"
      Dim value As String = "24-Jan-49"

      Dim cal As Calendar = CType(CultureInfo.CurrentCulture.Calendar.Clone(), Calendar)
      Console.WriteLine("Two Digit Year Range: {0} - {1}",
                        cal.TwoDigitYearMax - 99, cal.TwoDigitYearMax)

      Console.WriteLine("{0:d}", DateTime.ParseExact(value, fmt, Nothing))
      Console.WriteLine()

      cal.TwoDigitYearMax = 2099
      Dim culture As CultureInfo = CType(CultureInfo.CurrentCulture.Clone(),
CultureInfo)
      culture.DateTimeFormat.Calendar = cal
      Thread.CurrentThread.CurrentCulture = culture
```

```
        Console.WriteLine("Two Digit Year Range: {0} - {1}",
                          cal.TwoDigitYearMax - 99, cal.TwoDigitYearMax)
        Console.WriteLine("{0:d}", DateTime.ParseExact(value, fmt, Nothing))
    End Sub
End Module
' The example displays the following output:
'       Two Digit Year Range: 1930 - 2029
'       1/24/1949
'
'       Two Digit Year Range: 2000 - 2099
'       1/24/2049
```

The following example includes the "yy" custom format specifier in a custom format string.

VB

```
Dim date1 As Date = #12/1/0001#
Dim date2 As Date = #1/1/2010#
Console.WriteLine(date1.ToString("%y"))
' Displays 1
Console.WriteLine(date1.ToString("yy"))
' Displays 01
Console.WriteLine(date1.ToString("yyy"))
' Displays 001
Console.WriteLine(date1.ToString("yyyy"))
' Displays 0001
Console.WriteLine(date1.ToString("yyyyy"))
' Displays 00001
Console.WriteLine(date2.ToString("%y"))
' Displays 10
Console.WriteLine(date2.ToString("yy"))
' Displays 10
Console.WriteLine(date2.ToString("yyy"))
' Displays 2010
Console.WriteLine(date2.ToString("yyyy"))
' Displays 2010
Console.WriteLine(date2.ToString("yyyyy"))
' Displays 02010
```

Back to table

The "yyy" custom format specifier

The "yyy" custom format specifier represents the year with a minimum of three digits. If the year has more than three significant digits, they are included in the result string. If the year has fewer than three digits, the number is padded with leading zeros to produce three digits.

Note

For the Thai Buddhist calendar, which can have five-digit years, this format specifier displays all significant digits.

The following example includes the "yyy" custom format specifier in a custom format string.

VB

```
Dim date1 As Date = #12/1/0001#
Dim date2 As Date = #1/1/2010#
Console.WriteLine(date1.ToString("%y"))
' Displays 1
Console.WriteLine(date1.ToString("yy"))
' Displays 01
Console.WriteLine(date1.ToString("yyy"))
' Displays 001
Console.WriteLine(date1.ToString("yyyy"))
' Displays 0001
Console.WriteLine(date1.ToString("yyyyy"))
' Displays 00001
Console.WriteLine(date2.ToString("%y"))
' Displays 10
Console.WriteLine(date2.ToString("yy"))
' Displays 10
Console.WriteLine(date2.ToString("yyy"))
' Displays 2010
Console.WriteLine(date2.ToString("yyyy"))
' Displays 2010
Console.WriteLine(date2.ToString("yyyyy"))
' Displays 02010
```

Back to table

The "yyyy" custom format specifier

The "yyyy" custom format specifier represents the year with a minimum of four digits. If the year has more than four significant digits, they are included in the result string. If the year has fewer than four digits, the number is padded with leading zeros to produce four digits.

Note

For the Thai Buddhist calendar, which can have five-digit years, this format specifier displays a minimum of four digits.

The following example includes the "yyyy" custom format specifier in a custom format string.

VB

```
Dim date1 As Date = #12/1/0001#
Dim date2 As Date = #1/1/2010#
```

```
Console.WriteLine(date1.ToString("%y"))
' Displays 1
Console.WriteLine(date1.ToString("yy"))
' Displays 01
Console.WriteLine(date1.ToString("yyy"))
' Displays 001
Console.WriteLine(date1.ToString("yyyy"))
' Displays 0001
Console.WriteLine(date1.ToString("yyyyy"))
' Displays 00001
Console.WriteLine(date2.ToString("%y"))
' Displays 10
Console.WriteLine(date2.ToString("yy"))
' Displays 10
Console.WriteLine(date2.ToString("yyy"))
' Displays 2010
Console.WriteLine(date2.ToString("yyyy"))
' Displays 2010
Console.WriteLine(date2.ToString("yyyyy"))
' Displays 02010
```

Back to table

The "yyyyy" custom format specifier

The "yyyyy" custom format specifier (plus any number of additional "y" specifiers) represents the year with a minimum of five digits. If the year has more than five significant digits, they are included in the result string. If the year has fewer than five digits, the number is padded with leading zeros to produce five digits.

If there are additional "y" specifiers, the number is padded with as many leading zeros as necessary to produce the number of "y" specifiers.

The following example includes the "yyyyy" custom format specifier in a custom format string.

VB

```
Dim date1 As Date = #12/1/0001#
Dim date2 As Date = #1/1/2010#
Console.WriteLine(date1.ToString("%y"))
' Displays 1
Console.WriteLine(date1.ToString("yy"))
' Displays 01
Console.WriteLine(date1.ToString("yyy"))
' Displays 001
Console.WriteLine(date1.ToString("yyyy"))
' Displays 0001
Console.WriteLine(date1.ToString("yyyyy"))
' Displays 00001
Console.WriteLine(date2.ToString("%y"))
' Displays 10
Console.WriteLine(date2.ToString("yy"))
```

```
' Displays 10
Console.WriteLine(date2.ToString("yyy"))
' Displays 2010
Console.WriteLine(date2.ToString("yyyy"))
' Displays 2010
Console.WriteLine(date2.ToString("yyyyy"))
' Displays 02010
```

Back to table

The "z" custom format specifier

With DateTime values, the "z" custom format specifier represents the signed offset of the local operating system's time zone from Coordinated Universal Time (UTC), measured in hours. It does not reflect the value of an instance's DateTime.Kind property. For this reason, the "z" format specifier is not recommended for use with DateTime values.

With DateTimeOffset values, this format specifier represents the DateTimeOffset value's offset from UTC in hours.

The offset is always displayed with a leading sign. A plus sign (+) indicates hours ahead of UTC, and a minus sign (-) indicates hours behind UTC. A single-digit offset is formatted without a leading zero.

If the "z" format specifier is used without other custom format specifiers, it is interpreted as a standard date and time format specifier and throws a FormatException. For more information about using a single format specifier, see Using Single Custom Format Specifiers later in this topic.

The following example includes the "z" custom format specifier in a custom format string.

VB

```
Dim date1 As Date = Date.UtcNow
Console.WriteLine(String.Format("{0:%z}, {0:zz}, {0:zzz}", _
                  date1))
' Displays -7, -07, -07:00

Dim date2 As New DateTimeOffset(2008, 8, 1, 0, 0, 0, _
                                New Timespan(6, 0, 0))
Console.WriteLine(String.Format("{0:%z}, {0:zz}, {0:zzz}", _
                  date2))
' Displays +6, +06, +06:00
```

Back to table

The "zz" custom format specifier

With DateTime values, the "zz" custom format specifier represents the signed offset of the local operating system's time zone from UTC, measured in hours. It does not reflect the value of an instance's DateTime.Kind property. For this reason, the "zz" format specifier is not recommended for use with DateTime values.

With DateTimeOffset values, this format specifier represents the DateTimeOffset value's offset from UTC in hours.

The offset is always displayed with a leading sign. A plus sign (+) indicates hours ahead of UTC, and a minus sign (-) indicates hours behind UTC. A single-digit offset is formatted with a leading zero.

The following example includes the "zz" custom format specifier in a custom format string.

VB

```
Dim date1 As Date = Date.UtcNow
Console.WriteLine(String.Format("{0:%z}, {0:zz}, {0:zzz}", _
                  date1))
' Displays -7, -07, -07:00

Dim date2 As New DateTimeOffset(2008, 8, 1, 0, 0, 0, _
                                New Timespan(6, 0, 0))
Console.WriteLine(String.Format("{0:%z}, {0:zz}, {0:zzz}", _
                  date2))
' Displays +6, +06, +06:00
```

Back to table

The "zzz" custom format specifier

With DateTime values, the "zzz" custom format specifier represents the signed offset of the local operating system's time zone from UTC, measured in hours and minutes. It does not reflect the value of an instance's DateTime.Kind property. For this reason, the "zzz" format specifier is not recommended for use with DateTime values.

With DateTimeOffset values, this format specifier represents the DateTimeOffset value's offset from UTC in hours and minutes.

The offset is always displayed with a leading sign. A plus sign (+) indicates hours ahead of UTC, and a minus sign (-) indicates hours behind UTC. A single-digit offset is formatted with a leading zero.

The following example includes the "zzz" custom format specifier in a custom format string.

VB

```
Dim date1 As Date = Date.UtcNow
Console.WriteLine(String.Format("{0:%z}, {0:zz}, {0:zzz}", _
                  date1))
' Displays -7, -07, -07:00

Dim date2 As New DateTimeOffset(2008, 8, 1, 0, 0, 0, _
                                New Timespan(6, 0, 0))
Console.WriteLine(String.Format("{0:%z}, {0:zz}, {0:zzz}", _
                  date2))
' Displays +6, +06, +06:00
```

Back to table

The ":" custom format specifier

The ":" custom format specifier represents the time separator, which is used to differentiate hours, minutes, and seconds. The appropriate localized time separator is retrieved from the DateTimeFormatInfo.TimeSeparator property of the current or specified culture.

Note

To change the time separator for a particular date and time string, specify the separator character within a literal string delimiter. For example, the custom format string `hh'_'dd'_'ss` produces a result string in which "_" (an underscore) is always used as the time separator. To change the time separator for all dates for a culture, either change the value of the DateTimeFormatInfo.TimeSeparator property of the current culture, or instantiate a DateTimeFormatInfo object, assign the character to its TimeSeparator property, and call an overload of the formatting method that includes an IFormatProvider parameter.

If the ":" format specifier is used without other custom format specifiers, it is interpreted as a standard date and time format specifier and throws a FormatException. For more information about using a single format specifier, see Using Single Custom Format Specifiers later in this topic.

Back to table

The "/" custom format specifier

The "/" custom format specifier represents the date separator, which is used to differentiate years, months, and days. The appropriate localized date separator is retrieved from the DateTimeFormatInfo.DateSeparator property of the current or specified culture.

Note

To change the date separator for a particular date and time string, specify the separator character within a literal string delimiter. For example, the custom format string `mm'/'dd'/'yyyy` produces a result string in which "/" is always used as the date separator. To change the date separator for all dates for a culture, either change the value of the DateTimeFormatInfo.DateSeparator property of the current culture, or instantiate a DateTimeFormatInfo object, assign the character to its DateSeparator property, and call an overload of the formatting method that includes an IFormatProvider parameter.

If the "/" format specifier is used without other custom format specifiers, it is interpreted as a standard date and time format specifier and throws a FormatException. For more information about using a single format specifier, see Using Single Custom Format Specifiers later in this topic.

Back to table

Character literals

The following characters in a custom date and time format string are reserved and are always interpreted as formatting characters or, in the case of ", ', /, and \, as special characters.

F	H	K	M	d
f	g	h	m	s
t	y	z	%	:
/	"	'	\	

All other characters are always interpreted as character literals and, in a formatting operation, are included in the result string unchanged. In a parsing operation, they must match the characters in the input string exactly; the comparison is case-sensitive.

The following example includes the literal characters "PST" (for Pacific Standard Time) and "PDT" (for Pacific Daylight Time) to represent the local time zone in a format string. Note that the string is included in the result string, and that a string that includes the local time zone string also parses successfully.

VB

```
Imports System.Globalization
```

```
Module Example
   Public Sub Main()
      Dim formats() As String = { "dd MMM yyyy hh:mm tt PST",
                                  "dd MMM yyyy hh:mm tt PDT" }
      Dim dat As New Date(2016, 8, 18, 16, 50, 0)
      ' Display the result string.
      Console.WriteLine(dat.ToString(formats(1)))

      ' Parse a string.
      Dim value As String = "25 Dec 2016 12:00 pm PST"
      Dim newDate As Date
      If Date.TryParseExact(value, formats, Nothing,
                            DateTimeStyles.None, newDate) Then
         Console.WriteLine(newDate)
      Else
         Console.WriteLine("Unable to parse '{0}'", value)
      End If
   End Sub
End Module
' The example displays the following output:
'       18 Aug 2016 04:50 PM PDT
'       12/25/2016 12:00:00 PM
```

There are two ways to indicate that characters are to be interpreted as literal characters and not as reserve characters, so that they can be included in a result string or successfully parsed in an input string:

- By escaping each reserved character. For more information, see Using the Escape Character.

  The following example includes the literal characters "pst" (for Pacific Standard time) to represent the local time zone in a format string. Because both "s" and "t" are custom format strings, both characters must be escaped to be interpreted as character literals.

  VB

```
Imports System.Globalization

Module Example
   Public Sub Main()
      Dim fmt As String = "dd MMM yyyy hh:mm tt p\s\t"
      Dim dat As New Date(2016, 8, 18, 16, 50, 0)
      ' Display the result string.
      Console.WriteLine(dat.ToString(fmt))

      ' Parse a string.
      Dim value As String = "25 Dec 2016 12:00 pm pst"
      Dim newDate As Date
      If Date.TryParseExact(value, fmt, Nothing,
                            DateTimeStyles.None, newDate) Then
         Console.WriteLine(newDate)
      Else
         Console.WriteLine("Unable to parse '{0}'", value)
      End If
```

```
    End Sub
End Module
' The example displays the following output:
'       18 Aug 2016 04:50 PM pst
'       12/25/2016 12:00:00 PM
```

- By enclosing the entire literal string in quotation marks or apostrophes. The following example is like the previous one, except that "pst" is enclosed in quotation marks to indicate that the entire delimited string should be interpreted as character literals.

VB

```
Imports System.Globalization

Module Example
   Public Sub Main()
      Dim fmt As String = "dd MMM yyyy hh:mm tt ""pst"""
      Dim dat As New Date(2016, 8, 18, 16, 50, 0)
      ' Display the result string.
      Console.WriteLine(dat.ToString(fmt))

      ' Parse a string.
      Dim value As String = "25 Dec 2016 12:00 pm pst"
      Dim newDate As Date
      If Date.TryParseExact(value, fmt, Nothing,
                            DateTimeStyles.None, newDate) Then
         Console.WriteLine(newDate)
      Else
         Console.WriteLine("Unable to parse '{0}'", value)
      End If
   End Sub
End Module
' The example displays the following output:
'       18 Aug 2016 04:50 PM pst
'       12/25/2016 12:00:00 PM
```

Notes

Using single custom format specifiers

A custom date and time format string consists of two or more characters. Date and time formatting methods interpret any single-character string as a standard date and time format string. If they do not recognize the character as a valid format specifier, they throw a FormatException. For example, a format string that consists only of the specifier "h" is interpreted as a standard date and time format string. However, in this particular case, an exception is thrown because there is no "h" standard date and time format specifier.

To use any of the custom date and time format specifiers as the only specifier in a format string (that is, to use the "d", "f", "F", "g", "h", "H", "K", "m", "M", "s", "t", "y", "z", ":", or "/" custom format specifier by itself), include a space before or after the specifier, or include a percent ("%") format specifier before the single custom date and time specifier.

For example, "%h" is interpreted as a custom date and time format string that displays the hour represented by the current date and time value. You can also use the " h" or "h " format string, although this includes a space in the result string along with the hour. The following example illustrates these three format strings.

VB

```
Dim dat1 As Date = #6/15/2009 1:45PM#

Console.WriteLine("'{0:%h}'", dat1)
Console.WriteLine("'{0: h}'", dat1)
Console.WriteLine("'{0:h }'", dat1)
' The example displays the following output:
'       '1'
'       ' 1'
'       '1 '
```

Using the Escape Character

The "d", "f", "F", "g", "h", "H", "K", "m", "M", "s", "t", "y", "z", ":", or "/" characters in a format string are interpreted as custom format specifiers rather than as literal characters. To prevent a character from being interpreted as a format specifier, you can precede it with a backslash (\), which is the escape character. The escape character signifies that the following character is a character literal that should be included in the result string unchanged.

To include a backslash in a result string, you must escape it with another backslash (\\).

Note

Some compilers, such as the C++ and C# compilers, may also interpret a single backslash character as an escape character. To ensure that a string is interpreted correctly when formatting, you can use the verbatim string literal character (the @ character) before the string in C#, or add another backslash character before each backslash in C# and C++. The following C# example illustrates both approaches.

The following example uses the escape character to prevent the formatting operation from interpreting the "h" and "m" characters as format specifiers.

VB

```
Dim date1 As Date = #6/15/2009 13:45#
Dim fmt As String = "h \h m \m"

Console.WriteLine("{0} ({1}) -> {2}", date1, fmt, date1.ToString(fmt))
' The example displays the following output:
'       6/15/2009 1:45:00 PM (h \h m \m) -> 1 h 45 m
```

Control Panel settings

The **Regional and Language Options** settings in Control Panel influence the result string produced by a formatting operation that includes many of the custom date and time format specifiers. These settings are used to initialize the DateTimeFormatInfo object associated with the current thread culture, which provides values used to govern formatting. Computers that use different settings generate different result strings.

In addition, if you use the CultureInfo.CultureInfo(String) constructor to instantiate a new CultureInfo object that represents the same culture as the current system culture, any customizations established by the **Regional and Language Options** item in Control Panel will be applied to the new CultureInfo object. You can use the CultureInfo.CultureInfo(String, Boolean) constructo to create a CultureInfo object that does not reflect a system's customizations.

DateTimeFormatInfo properties

Formatting is influenced by properties of the current DateTimeFormatInfo object, which is provided implicitly by the current thread culture or explicitly by the IFormatProvider parameter of the method that invokes formatting. For the IFormatProvider parameter, you should specify a CultureInfo object, which represents a culture, or a DateTimeFormatInfo object.

The result string produced by many of the custom date and time format specifiers also depends on properties of the current DateTimeFormatInfo object. Your application can change the result produced by some custom date and time format specifiers by changing the corresponding DateTimeFormatInfo property. For example, the "ddd" format specifier adds an abbreviated weekday name found in the AbbreviatedDayNames string array to the result string. Similarly, the "MMMM" format specifier adds a full month name found in the MonthNames string array to the result string.

See Also

System.DateTime
System.IFormatProvider
Formatting Types in the .NET Framework
Standard Date and Time Format Strings
Sample: .NET Framework 4 Formatting Utility