.NET Standard DateTime Format Strings

Specifier	Name	Description
`d`	Short date pattern	Represents a custom DateTime format string defined by the current ShortDatePattern property.
`D`	Long date pattern	Represents a custom DateTime format string defined by the current LongDatePattern property.
`f`	Full date/time pattern (short time)	Represents a combination of the long date (D) and short time (t) patterns, separated by a space.
`F`	Full date/time pattern (long time)	Represents a custom DateTime format string defined by the current FullDateTimePattern property.
`g`	General date/time pattern (short time)	Represents a combination of the short date (d) and short time (t) patterns, separated by a space.
`G`	General date/time pattern (long time)	Represents a combination of the short date (d) and long time (T) patterns, separated by a space.
`M` or `m`	Month day pattern	Represents a custom DateTime format string defined by the current MonthDayPattern property.
`o`	Round-trip date/time pattern	Represents a custom DateTime format string using a pattern that preserves time zone information. The pattern is designed to round-trip DateTime formats, including the Kind property, in text. Then the formatted string can be parsed back using Parse or ParseExact with the correct Kind property value. Equivalent custom format string is "yyyy'-'MM'-'dd'T'HH':'mm':'ss.fffffffK".
`R` or `r`	RFC1123 pattern	Represents a custom DateTime format string defined by the current RFC1123Pattern property. The pattern is a defined standard and the property is read-only. Equivalent custom format string is "ddd, dd MMM yyyy HH':'mm':'ss 'GMT'". Does not convert DateTimes to UTC.
`s`	Sortable date/time pattern; ISO 8601	Represents a custom DateTime format string defined by the current SortableDateTimePattern property. This pattern is a defined standard and the property is read-only. Equivalent custom format string is "yyyy'-'MM'-'dd'T'HH':'mm':'ss".
`t`	Short time pattern	Represents a custom DateTime format string defined by the current ShortTimePattern property. For example, the custom format string for the invariant culture is "HH:mm".
`T`	Long time pattern	Represents a custom DateTime format string defined by the current LongTimePattern property. For example, the custom format string for the invariant culture is "HH:mm:ss".
`u`	Universal sortable date/time pattern	Represents a custom DateTime format string defined by the current UniversalSortableDateTimePattern property. Equivalent custom format string is "yyyy'-'MM'-'dd HH':'mm':'ss'Z'". Does not convert DateTimes to UTC.
`U`	Universal sortable date/time pattern	Represents a custom DateTime format string defined by the current FullDateTimePattern property. This pattern is the same as the full date/long time (F) pattern. However, formatting operates on the Coordinated Universal Time (UTC) that is equivalent to the DateTime object being formatted.
`Y` or `y`	Year month pattern	Represents a custom DateTime format string defined by the current YearMonthPattern property. For example, the custom format string for the invariant culture is "yyyy MMMM".
Any other single character	(Unknown specifier)	An unknown specifier throws a runtime format exception.

.NET Custom DateTime Format Strings

Specifier	Description
d	Represents the day of the month as a number from 1 through 31. A single-digit day is formatted without a leading zero.
dd	Represents the day of the month as a number from 01 through 31. A single-digit day is formatted with a leading zero.
ddd	Represents the abbreviated name of the day of the week as defined in the current System.Globalization.DateTimeFormatInfo.AbbreviatedDayNames property.
dddd	Represents the full name of the day of the week as defined in the current System.Globalization.DateTimeFormatInfo.DayNames property.
f	Represents the most significant digit of the seconds fraction. Note that if the "f" format specifier is used alone, without other format specifiers, it is interpreted as the "f" standard DateTime format specifier (full date/time pattern). When you use this format specifier with the ParseExact or TryParseExact method, the number of "f" format specifiers that you use indicates the number of most significant digits of the seconds fraction to parse.
ff…	Number of repeated specifiers represents most significant digits of the seconds fraction.
F	Represents the most significant digit of the seconds fraction. Nothing is displayed if the digit is zero. When you use this format specifier with the ParseExact or TryParseExact method, the number of "F" format specifiers that you use indicates the maximum number of most significant digits of the seconds fraction to parse.
FF…	Number of repeated specifiers represents most significant digits of the seconds fraction. Trailing zeros, or two zero digits, are not displayed.
g or **gg**	Represents the period or era (A.D. for example). This specifier is ignored if the date to be formatted does not have an associated period or era string.
h	Represents the hour as a number from 1 through 12, that is, the hour as represented by a 12-hour clock that counts the whole hours since midnight or noon. A single-digit hour is formatted without a leading zero.
hh	Represents the hour as a number from 01 through 12, that is, the hour as represented by a 12-hour clock that counts the whole hours since midnight or noon. A single-digit hour is formatted with a leading zero.
H	Represents the hour as a number from 0 through 23, that is, the hour as represented by a zero-based 24-hour clock that counts the hours since midnight. A single-digit hour is formatted without a leading zero.
HH	Represents the hour as a number from 00 through 23, that is, the hour as represented by a zero-based 24-hour clock that counts the hours since midnight. A single-digit hour is formatted with a leading zero.
K	Represents different values of the DateTime.Kind property, that is, Local, Utc, or Unspecified. This specifier round-trips the kind value in text and preserves the time zone. For the Local kind value, this specifier is equivalent to the "zzz" specifier and displays the local offset, for example, "-07:00". For the Utc kind value, the specifier displays a "Z" character to represent a UTC date. For the Unspecified kind value, the specifier is equivalent to "" (nothing).
m	Represents the minute as a number from 0 through 59. The minute represents whole minutes passed since the last hour. A single-digit minute is formatted without a leading zero.
mm	Represents the minute as a number from 00 through 59. The minute represents whole minutes passed since the last hour. A single-digit minute is formatted with a leading zero.
M	Represents the month as a number from 1 through 12. A single-digit month is formatted without a leading zero.
MM	Represents the month as a number from 01 through 12. A single-digit month is formatted with a leading zero.
MMM	Represents the abbreviated name of the month as defined in the current System.Globalization.DateTimeFormatInfo.AbbreviatedMonthNames property.
MMMM	Represents the full name of the month as defined in the current System.Globalization.DateTimeFormatInfo.MonthNames property.

s	Represents the seconds as a number from 0 through 59. The second represents whole seconds passed since the last minute. A single-digit second is formatted without a leading zero.
ss	Represents the seconds as a number from 00 through 59. The second represents whole seconds passed since the last minute. A single-digit second is formatted with a leading zero.
t	Represents the first character of the A.M./P.M. designator defined in the current System.Globalization.DateTimeFormatInfo.AMDesignator or System.Globalization.DateTimeFormatInfo.PMDesignator property.
tt	Represents the A.M./P.M. designator as defined in the current System.Globalization.DateTimeFormatInfo.AMDesignator or System.Globalization.DateTimeFormatInfo.PMDesignator property.
y	Represents the year as at most a two-digit number. If the year has more than two digits, only the two low-order digits appear in the result. If the year has fewer than two digits, the number is formatted without a leading zero.
yy	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 year has fewer than two digits, the number is padded with leading zeroes to achieve two digits.
yyy	Represents the year as a three-digit number. If the year has more than three digits, only the three low-order digits appear in the result. If the year has fewer than three digits, the number is padded with leading zeroes to achieve three digits. Note that for the Thai Buddhist calendar, which can have five-digit years, this format specifier displays all five digits.
yyyy	Represents the year as a four-digit number. If the year has more than four digits, only the four low-order digits appear in the result. If the year has fewer than four digits, the number is padded with leading zeroes to achieve four digits. Note that for the Thai Buddhist calendar, which can have five-digit years, this format specifier renders all five digits.
yyyyy	Represents the year as a five-digit number. If the year has more than five digits, only the five low-order digits appear in the result. If the year has fewer than five digits, the number is padded with leading zeroes to achieve five digits. If there are additional "y" specifiers, the number is padded with as many leading zeroes as necessary to achieve the number of "y" specifiers.
z	Represents the signed time zone offset of your system from Greenwich Mean Time (GMT) measured in hours. For example, the offset for a computer in the Pacific Standard Time zone is "-8". The offset is always displayed with a leading sign. A plus sign (+) indicates hours ahead of GMT and a minus sign (-) indicates hours behind GMT. The offset ranges from –12 through +13. A single-digit offset is formatted without a leading zero. The offset is affected by daylight savings time.
zz	Represents the signed time zone offset of your system from Greenwich Mean Time (GMT) measured in hours. For example, the offset for a computer in the Pacific Standard Time zone is "-08". The offset is always displayed with a leading sign. A plus sign (+) indicates hours ahead of GMT and a minus sign (-) indicates hours behind GMT. The offset ranges from –12 through +13. A single-digit offset is formatted with a leading zero. The offset is affected by daylight savings time.
zzz	Represents the signed time zone offset of your system from Greenwich Mean Time (GMT) measured in hours and minutes. For example, the offset for a computer in the Pacific Standard Time zone is "-08:00". The offset is always displayed with a leading sign. A plus sign (+) indicates hours ahead of GMT and a minus sign (-) indicates hours behind GMT. The offset ranges from –12 through +13. A single-digit offset is formatted with a leading zero. The offset is affected by daylight savings time.
:	The time separator defined in the current System.Globalization.DateTimeFormatInfo.TimeSeparator property that is used to differentiate hours, minutes, and seconds.
/	The date separator defined in the current System.Globalization.DateTimeFormatInfo.DateSeparator property that is used to differentiate years, months, and days.
"	Quoted string (quotation mark). Displays the literal value of any string between two quotation marks ("). Precede each quotation mark with an escape character (\).
'	Quoted string (apostrophe). Displays the literal value of any string between two apostrophe (') characters.
%c	Represents the result associated with a custom format specifier "c", when the custom DateTime format string consists solely of that custom format specifier. That is, to use the "d", "f", "F", "h", "m", "s", "t", "y", "z", "H", or "M" custom format specifier by itself, specify "%d", "%f", "%F", "%h", "%m", "%s", "%t", "%y", "%z", "%H", or "%M".
\c	The escape character. Displays the character "c" as a literal when that character is preceded by the escape character (\). To insert the backslash character itself in the result string, use two escape characters ("\\").
Any other char.	Any other character is copied to the result string, and does not affect formatting.

.NET Standard Number Format Strings

Specifier	Name	Description
C or **c**	Currency	The number is converted to a string that represents a currency amount. The conversion is controlled by the currency format information of the current NumberFormatInfo object. Precision specifier (eg. "{0:C5}" allowed.
D or **d**	Decimal	This format is supported only for integral types. The number is converted to a string of decimal digits (0-9), prefixed by a minus sign if the number is negative. Precision specifier (eg. "{0:d3}" allowed.
E or **e**	Scientific (exponential)	The number is converted to a string of the form "-d.ddd…E+ddd" or "-d.ddd…e+ddd", where each 'd' indicates a digit (0-9). The string starts with a minus sign if the number is negative. One digit always precedes the decimal point. Precision specifier (eg. "{0:E5}" allowed. The case of the format specifier indicates whether to prefix the exponent with an 'E' or an 'e'. The exponent always consists of a plus or minus sign and a minimum of three digits. The exponent is padded with zeros to meet this minimum, if required.
F or **f**	Fixed-point	The number is converted to a string of the form "-ddd.ddd…" where each 'd' indicates a digit (0-9). The string starts with a minus sign if the number is negative. Precision specifier (eg. "{0:f4}" allowed.
G or **g**	General	The number is converted to the most compact of either fixed-point or scientific notation, depending on the type of the number and whether a precision specifier is present.
N or **n**	Number	The number is converted to a string of the form "-d,ddd,ddd.ddd…", where '-' indicates a negative number symbol if required, 'd' indicates a digit (0-9), ',' indicates a thousand separator between number groups, and '.' indicates a decimal point symbol. The actual negative number pattern, number group size, thousand separator, and decimal separator are specified by the current NumberFormatInfo object. Precision specifier (eg. "{0:N5}" allowed.
P or **p**	Percent	The number is converted to a string that represents a percent as defined by the NumberFormatInfo.PercentNegativePattern property if the number is negative, or the NumberFormatInfo.PercentPositivePattern property if the number is positive. The converted number is multiplied by 100 in order to be presented as a percentage. Precision specifier (eg. "{0:p6}" allowed.
R or **r**	Round-trip	This format is supported only for the Single and Double types. The round-trip specifier guarantees that a numeric value converted to a string will be parsed back into the same numeric value. When a numeric value is formatted using this specifier, it is first tested using the general format, with 15 spaces of precision for a Double and 7 spaces of precision for a Single. If the value is successfully parsed back to the same numeric value, it is formatted using the general format specifier. However, if the value is not successfully parsed back to the same numeric value, then the value is formatted using 17 digits of precision for a Double and 9 digits of precision for a Single. Precision specifier NOT allowed.
X or **x**	Hexadecimal	This format is supported only for integral types. The number is converted to a string of hexadecimal digits. The case of the format specifier indicates whether to use uppercase or lowercase characters for the hexadecimal digits greater than 9. Precision specifier (eg. "{0:x4}" allowed. If required, the number is padded with zeros to its left to produce the number of digits given by the precision specifier.
Any other single char.	(Unknown specifier)	An unknown specifier throws a runtime format exception.

Examples (en-US)

Format String	Data type	Value	Output
C	Double	12345.6789	$12,345.68
D	Int32	12345	12345
D8	Int32	12345	00012345
E	Double	12345.6789	1.234568E+004
E10	Double	12345.6789	1.2345678900E+004
E	Double	12345.6789	1.2346e+004

Format String	Data type	Value	Output
F	Double	12345.6789	12345.68
F0	Double	12345.6789	123456
F6	Double	12345.6789	12345.678900
G	Double	12345.6789	12345.6789
G7	Double	12345.6789	12345.68
G	Double	0.0000023	2.3E-6

Format String	Data type	Value	Output
G2	Double	1234	1.2E3
G	Double	Math.PI	3.14159265358979
N	Double	12345.6789	12,345.68
N4	Double	123456789	123,456,789.0000
P	Double	.126	12.60 %
r	Double	Math.PI	3.141592653589793

.NET Custom Number Format Strings

Specifier	Name	Description
0	Zero placeholder	If the value being formatted has a digit in the position where the '0' appears in the format string, then that digit is copied to the result string. The position of the leftmost '0' before the decimal point and the rightmost '0' after the decimal point determines the range of digits that are always present in the result string. The "00" specifier causes the value to be rounded to the nearest digit preceding the decimal, where rounding away from zero is always used. For example, formatting 34.5 with "00" would result in the value 35.
#	Digit placeholder	If the value being formatted has a digit in the position where the '#' appears in the format string, then that digit is copied to the result string. Otherwise, nothing is stored in that position in the result string. This specifier never displays the '0' character if it is not a significant digit, even if '0' is the only digit in the string. It will display the '0' character if it is a significant digit in the number being displayed. The "##" format string causes the value to be rounded to the nearest digit preceding the decimal, where rounding away from zero is always used. For example, formatting 34.5 with "##" would result in the value 35.
.	Decimal point	The first '.' character in the format string determines the location of the decimal separator in the formatted value; any additional '.' characters are ignored. The actual character used as the decimal separator is determined by the NumberDecimalSeparator property of the NumberFormatInfo that controls formatting.
,	Thousand separator and number scaling	**Thousand Separator Specifier** If one or more ',' characters is specified between two digit placeholders (0 or #) that format the integral digits of a number, a group separator character is inserted between each number group in the integral part of the output. The NumberGroupSeparator and NumberGroupSizes properties of the current NumberFormatInfo object determine the character used as the number group separator and the size of each number group. **Number Scaling Specifier** If one or more ',' characters is specified immediately to the left of the explicit or implicit decimal point, the number to be formatted is divided by 1000 each time a number scaling specifier occurs. For example, if the string "0,," is used to format the number 100 million, the output is "100". You can use thousand separator and number scaling specifiers in the same format string.
%	Percentage placeholder	The presence of a '%' character in a format string causes a number to be multiplied by 100 before it is formatted. The appropriate symbol is inserted in the number itself at the location where the '%' appears in the format string. The percent character used is dependent on the current NumberFormatInfo class.
E0 **E+0** **E-0** **e0** **e+0** **e-0**	Scientific notation	If any of the strings "E", "E+", "E-", "e", "e+", or "e-" are present in the format string and are followed immediately by at least one '0' character, then the number is formatted using scientific notation with an 'E' or 'e' inserted between the number and the exponent. The number of '0' characters following the scientific notation indicator determines the minimum number of digits to output for the exponent. The "E+" and "e+" formats indicate that a sign character (plus or minus) should always precede the exponent. The "E", "E-", "e", or "e-" formats indicate that a sign character should only precede negative exponents.
****	Escape character	In C# and C++, the backslash character causes the next character in the format string to be interpreted as an escape sequence. It is used with traditional formatting sequences like '\n' (new line). In some languages, the escape character itself must be preceded by an escape character when used as a literal. Otherwise, the compiler interprets the character as an escape sequence. Use the string "\\" to display '\'. Note that this escape character is not supported in Visual Basic; however, ControlChars provides the same functionality.
'ABC' **"ABC"**	Literal string	Characters enclosed in single or double quotes are copied to the result string, and do not affect formatting.
;	Section separator	The ';' character is used to separate sections for positive, negative, and zero numbers in the format string.
Other	All other characters	Any other character is copied to the result string, and does not affect formatting.

Section Separators and Conditional Formatting

Different formatting can be applied to a string based on whether the value is positive, negative, or zero. To produce this behavior, a custom format string can contain up to three sections separated by semicolons. These sections are described in the following table.

No. of Sections	Description
One section	The format string applies to all values.
Two sections	The first section applies to positive values and zeros, and the second section applies to negative values. If the number to be formatted is negative, but becomes zero after rounding according to the format in the second section, then the resulting zero is formatted according to the first section.
Three sections	The first section applies to positive values, the second section applies to negative values, and the third section applies to zeros. The second section can be left empty (by having nothing between the semicolons), in which case the first section applies to all nonzero values. If the number to be formatted is nonzero, but becomes zero after rounding according to the format in the first or second section, then the resulting zero is formatted according to the third section.

Section separators ignore any preexisting formatting associated with a number when the final value is formatted. For example, negative values are always displayed without a minus sign when section separators are used. If you want the final formatted value to have a minus sign, you should explicitly include the minus sign as part of the custom format specifier.

Custom Numeric Format Strings Output Examples

The following table illustrates the output created by applying some custom numeric format strings to specific data types and values. The output was generated using the ToString method and the English-United States (en-US) culture.

Format string	Data type	Value	Output
`#####`	Double	123	123
`00000`	Double	123	00123
`(###) ### - ####`	Double	1234567890	(123) 456 – 7890
`#.##`	Double	1.2	1.2
`0.00`	Double	1.2	1.20
`00.00`	Double	1.2	01.20
`#,#`	Double	1234567890	1,234,567,890
`#,,`	Double	1234567890	1235
`#,,,`	Double	1234567890	1
`#,##0,,`	Double	1234567890	1,235
`#0.##%`	Double	0.086	8.6%
`0.###E+0`	Double	86000	8.6E+4
`0.###E+000`	Double	86000	8.6E+004
`0.###E-000`	Double	86000	8.6E004
`[##-##-##]`	Double	123456	[12-34-56]
`##;(##)`	Double	1234	1234
`##;(##)`	Double	-1234	(1234)