CFont::CreateFont

BOOL CreateFont(int *nHeight*, **int** *nWidth*,
int *nEscapement*, **int** *nOrientation*, **int** *nWeight*,
BYTE *bItalic*, **BYTE** *bUnderline*, **BYTE** *cStrikeOut*,
BYTE *nCharSet*, **BYTE** *nOutPrecision*,
BYTE *nClipPrecision*, **BYTE** *nQuality*,
BYTE *nPitchAndFamily*,
LPCTSTR *lpszFacename* **);**

Return Value

Nonzero if successful; otherwise 0.

Parameters

nHeight

Specifies the desired height (in logical units) of the font. The font height can be specified in the following ways:

- Greater than 0, in which case the height is transformed into device units and matched against the cell height of the available fonts.
- Equal to 0, in which case a reasonable default size is used.
- Less than 0, in which case the height is transformed into device units and the absolute value is matched against the character height of the available fonts. The absolute value of *nHeight* must not exceed 16,384 device units after it is converted. For all height comparisons, the font mapper looks for the largest font that does not exceed the requested size or the smallest font if all the fonts exceed the requested size.

nWidth

Specifies the average width (in logical units) of characters in the font. If *nWidth* is 0, the aspect ratio of the device will be matched against the digitization aspect ratio of the available fonts to find the closest match, which is determined by the absolute value of the difference.

nEscapement

Specifies the angle (in 0.1-degree units) between the escapement vector and the x-axis of the display surface. The escapement vector is the line through the origins of the first and last characters on a line. The angle is measured counterclockwise from the x-axis.

nOrientation

Specifies the angle (in 0.1-degree units) between the baseline of a character and the x-axis. The angle is measured counterclockwise from the x-axis for coordinate systems in which the y-direction is down and clockwise from the x-axis for coordinate systems in which the y-direction is up.

nWeight

Specifies the font weight (in inked pixels per 1000). Although *nWeight* can be any integer value from 0 to 1000, the common constants and values are as follows:

Constant	Value
FW_DONTCARE	0
FW_THIN	100
FW_EXTRALIGHT	200
FW_ULTRALIGHT	200
FW_LIGHT	300
FW_NORMAL	400
FW_REGULAR	400
FW_MEDIUM	500
FW_SEMIBOLD	600
FW_DEMIBOLD	600
FW_BOLD	700
FW_EXTRABOLD	800
FW_ULTRABOLD	800

FW_BLACK	900
FW_HEAVY	900

These values are approximate; the actual appearance depends on the typeface. Some fonts have only **FW_NORMAL**, **FW_REGULAR**, and **FW_BOLD** weights. If **FW_DONTCARE** is specified, a default weight is used.

bItalic

Specifies whether the font is italic.

bUnderline

Specifies whether the font is underlined.

cStrikeOut

Specifies whether characters in the font are struck out. Specifies a strikeout font if set to a nonzero value.

nCharSet

Specifies the font's character set. The following constants and values are predefined:

Constant	Value
ANSI_CHARSET	0
DEFAULT_CHARSET	1
SYMBOL_CHARSET	2
SHIFTJIS_CHARSET	128
OEM_CHARSET	255

The OEM character set is system-dependent.

Fonts with other character sets may exist in the system. An application that uses a font with an unknown character set must not attempt to translate or interpret strings that are to be rendered with that font. Instead, the strings should be passed directly to the output device driver.

The font mapper does not use the **DEFAULT_CHARSET** value. An application can use this value to allow the name and size of a font to fully describe the logical font. If a font with the specified name does not exist, a font from any character set can be substituted for the specified font. To avoid unexpected results, applications should use the **DEFAULT_CHARSET** value sparingly.

nOutPrecision

Specifies the desired output precision. The output precision defines how closely the output must match the requested font's height, width, character orientation, escapement, and pitch. It can be any one of the following values:

OUT_CHARACTER_PRECIS	**OUT_STRING_PRECIS**
OUT_DEFAULT_PRECIS	**OUT_STROKE_PRECIS**
OUT_DEVICE_PRECIS	**OUT_TT_PRECIS**
OUT_RASTER_PRECIS	

Applications can use the **OUT_DEVICE_PRECIS**, **OUT_RASTER_PRECIS**, and **OUT_TT_PRECIS** values to control how the font mapper chooses a font when the system contains more than one font with a given name. For example, if a system contains a font named Symbol in raster and TrueType form, specifying **OUT_TT_PRECIS** forces the font mapper to choose the TrueType version. (Specifying **OUT_TT_PRECIS** forces the font mapper to choose a TrueType font whenever the specified font name matches a device or raster font, even when there is no TrueType font of the same name.)

nClipPrecision

Specifies the desired clipping precision. The clipping precision defines how to clip characters that are partially outside the clipping region. It can be any one of the following values:

CLIP_CHARACTER_PRECIS	**CLIP_MASK**
CLIP_DEFAULT_PRECIS	**CLIP_STROKE_PRECIS**
CLIP_ENCAPSULATE	**CLIP_TT_ALWAYS**
CLIP_LH_ANGLES	

To use an embedded read-only font, an application must specify **CLIP_ENCAPSULATE**. To achieve consistent rotation of device, TrueType, and vector fonts, an application can use the OR operator to combine the **CLIP_LH_ANGLES** value with any of the other *nClipPrecision* values. If the **CLIP_LH_ANGLES** bit is set, the rotation for all fonts

depends on whether the orientation of the coordinate system is left-handed or right-handed. (For more information about the orientation of coordinate systems, see the description of the *nOrientation* parameter.) If **CLIP_LH_ANGLES** is not set, device fonts always rotate counterclockwise, but the rotation of other fonts is dependent on the orientation of the coordinate system.

nQuality

Specifies the font's output quality, which defines how carefully the GDI must attempt to match the logical-font attributes to those of an actual physical font. It can be one of the following values:

- **DEFAULT_QUALITY** Appearance of the font does not matter.
- **DRAFT_QUALITY** Appearance of the font is less important than when **PROOF_QUALITY** is used. For GDI raster fonts, scaling is enabled. Bold, italic, underline, and strikeout fonts are synthesized if necessary.
- **PROOF_QUALITY** Character quality of the font is more important than exact matching of the logical-font attributes. For GDI raster fonts, scaling is disabled and the font closest in size is chosen. Bold, italic, underline, and strikeout fonts are synthesized if necessary.

nPitchAndFamily

Specifies the pitch and family of the font. The two low-order bits specify the pitch of the font and can be any one of the following values:

DEFAULT_PITCH	**VARIABLE_PITCH**	**FIXED_PITCH**

Applications can add **TMPF_TRUETYPE** to the *nPitchAndFamily* parameter to choose a TrueType font. The four high-order bits of the parameter specify the font family and can be any one of the following values:

- **FF_DECORATIVE** Novelty fonts: Old English, for example.
- **FF_DONTCARE** Don't care or don't know.
- **FF_MODERN** Fonts with constant stroke width (fixed-pitch), with or without serifs. Fixed-pitch fonts are usually modern faces. Pica, Elite, and Courier New are examples.
- **FF_ROMAN** Fonts with variable stroke width (proportionally spaced) and with serifs. Times New Roman and Century Schoolbook are examples.
- **FF_SCRIPT** Fonts designed to look like handwriting. Script and Cursive are examples.
- **FF_SWISS** Fonts with variable stroke width (proportionally spaced) and without serifs. MS Sans Serif is an example.
  An application can specify a value for *nPitchAndFamily* by using the Boolean OR operator to join a pitch constant with a family constant.
  Font families describe the look of a font in a general way. They are intended for specifying fonts when the exact typeface desired is not available.

lpszFacename

A **CString** or pointer to a null-terminated string that specifies the typeface name of the font. The length of this string must not exceed 30 characters. The Windows **EnumFontFamilies** function can be used to enumerate all currently available fonts. If *lpszFacename* is **NULL**, the GDI uses a device-independent typeface.

Remarks

Initializes a **CFont** object with the specified characteristics. The font can subsequently be selected as the font for any device context.

The **CreateFont** function does not create a new Windows GDI font. It merely selects the closest match from the fonts available in the GDI's pool of physical fonts.

Applications can use the default settings for most of these parameters when creating a logical font. The parameters that should always be given specific values are *nHeight* and *lpszFacename*. If *nHeight* and *lpszFacename* are not set by the application, the logical font that is created is device-dependent.

When you finish with the **CFont** object created by the **CreateFont** function, first select the font out of the device context, then delete the **CFont** object.

Beispiel:

```
CFont font

font.CreateFont( 45, 45,
  0,0,  FW_DONTCARE,                    oder FW_BOLD
  FALSE, FALSE, FALSE,
  DEFAULT_CHARSET,
  OUT_CHARACTER_PRECIS,
  CLIP_CHARACTER_PRECIS,
  DEFAULT_QUALITY ,
  DEFAULT_PITCH |  FF_DONTCARE ,
"Times Roman");

CFont* pOldFont;
pOldFont = dc.SelectObject( &font);

dc.TextOut(100,100, "Dies ist ein Text mit neuer Schrift");

dc.SelectObject( pOldFont );
```