

TextOut

The **TextOut** function writes a character string at the specified location, using the currently selected font.

```
BOOL TextOut(  
    HDC hdc,                // handle of device context  
    int nXStart,             // x-coordinate of starting position  
    int nYStart,             // y-coordinate of starting position  
    LPCTSTR lpString,       // address of string  
    int cbString            // number of characters in string  
);
```

Parameters

hdc

Identifies the device context.

nXStart

Specifies the logical x-coordinate of the reference point that Windows uses to align the string.

nYStart

Specifies the logical y-coordinate of the reference point that Windows uses to align the string.

lpString

Points to the string to be drawn. The string does not need to be zero-terminated, since *cbString* specifies the length of the string.

cbString

Specifies the number of characters in the string.

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE. To get extended error information, call [GetLastError](#).

Remarks

The interpretation of the reference point depends on the current text-alignment mode. An application can retrieve this mode by calling the [GetTextAlign](#) function; an application can alter this mode by calling the [SetTextAlign](#) function.

By default, the current position is not used or updated by this function. However, an application can call the **SetTextAlign** function with the *fMode* parameter set to TA_UPDATECP to permit Windows to use and update the current position each time the application calls **TextOut** for a specified device context. When this flag is set, Windows ignores the *nXStart* and *nYStart* parameters on subsequent **TextOut** calls.

When the **TextOut** function is placed inside a path bracket, the system generates a path for the TrueType text that includes each character plus its character box. The region generated is the character box minus the text, rather than the text itself. You can obtain the region enclosed by the outline of the TrueType text by setting the background mode to transparent before placing the **TextOut** function in the path bracket. Following is sample code that demonstrates this procedure.

```
// Obtain the window's client rectangle  
GetClientRect(hwnd, &r);  
  
// THE FIX: by setting the background mode  
// to transparent, the region is the text itself  
// SetBkMode(hdc, TRANSPARENT);  
  
// Bracket begin a path  
BeginPath(hdc);
```

```
// Send some text out into the world
TextOut(hdc, r.left, r.top, "Defenestration can be hazardous", 4);

// Bracket end a path
EndPath(hdc);

// Derive a region from that path
SelectClipPath(hdc, RGN_AND);

// This generates the same result as SelectClipPath()
// SelectClipRgn(hdc, PathToRegion(hdc));

// Fill the region with grayness
FillRect(hdc, &r, GetStockObject(GRAY_BRUSH));
```