# How do I display an RTL string in a notification balloon on an LTR system?

devblogs.microsoft.com/oldnewthing/20131219-00

December 19, 2013

Raymond Chen

Suppose you have a program that is written in Arabic or Hebrew and you want to render some text. No problem. You just call `ExtTextOut` and pass the `ETO_RTLREADING` flag to say, "Please render this string in an RTL context." Many other text-rendering functions have a similar flag, such as `DT_RTLREADING` for `DrawText`.

But what if you don't control the call to `ExtTextOut` or `DrawText` or whatever other function is being used to render the text. If you don't control the call, then you can't pass along the magic "Please render this string in an RTL context" flag.

If you're lucky, the component that is doing the rendering has some analogous flag that tells it to render in RTL context. If the component is a control, this flag may be implicit in the `WS_EX_RTLREADING` extended style on the control window itself. For some components, the secret signal is the presence of two RLM characters (U+200F) at the beginning of the string.

If you're not lucky, then the component that is doing the rendering gives you no way to convince or cajole it into rendering text in an RTL context. But all hope is not lost: The (possibly non-intuitive) Unicode Bidi algorithm comes to the rescue!

What you can do is place the RLE control character (U+202B) at the start of the string. The RIGHT-TO-LEFT EMBEDDING control character means "Treat the text that follows in an RTL context until further instructions." (You cancel the effect of an RLE by a PDF (POP DIRECTIONAL FORMATTING, U+202C).)

Let's demonstrate in our scratch program.

```
#define THESTRING L"\x0639\x0644\x0649 \x0633\x0628\x064a\x0644 " \
                  L"\x0627\x0644\x0645\x062b\x0627\x0644: " \
                  L"Dear \x0623\x0634\x0631\x0641 " \
                  L"\x0645\x0627\x0647\x0631"
#define RLE L"\x202b"
void ShowString(HDC hdc, int y, PCWSTR psz, UINT format)
{
 RECT rc = { 0, y, 500, y+100 };
 DrawTextW(hdc, psz, -1, &rc, format);
}
void
PaintContent(HWND hwnd, PAINTSTRUCT *pps)
{
 ShowString(pps->hdc, 0, THESTRING, 0);
 ShowString(pps->hdc, 100, THESTRING, DT_RTLREADING);
 ShowString(pps->hdc, 200, RLE THESTRING, 0);
}
```

This sample program takes a string in Arabic (with a little bit of English thrown in just to make the difference more noticeable) and renders it three ways:

- As an LTR string with no special formatting.
- As an RTL string with no special formatting.
- As an LTR string with an RTL context imposed via the RLE control character.

Observe that in the first case, the string treats the Arabic at the beginning and end of the string as Arabic text embedded in an English sentence, so it is formatted as

> على سبيل المثال: Dear أشرف ماهر

In the second case, the entire string is treated as an Arabic sentence with an English word stuck inside it. Therefore, it comes out as

> على سبيل المثال: Dear أشرف ماهر

In the third case, we force the string to be treated as an Arabic sentence by using the RLE control character. The result matches the second string.

Note that the formatting is still not ideal because the underlying canvas is still LTR: The text is left-justified instead of right-justified, and the caption buttons on the window will still be drawn in the LTR position. But it's better than nothing.

Raymond Chen

**Follow**