The code page on the server is not necessarily the code page on the client

devblogs.microsoft.com/oldnewthing/20070914-00

September 14, 2007

Raymond Chen

It's not enough to choose a code page. You have to choose the right code page.

> We have a system that reformats and reinstalls a network client computer each time it boots up. The client connects to the server to obtain a loader program, and the loader program then connects to the server to download the actual operating system. If anything goes wrong, the server sends an error message to the client, which is displayed on the screen while it's still in character mode. (No Unicode available here.)
>
> Initially, we used `FormatMessageA` to generate the error message, but somebody told us we should use `FormatMessageW` followed by `WideCharToMultiByte(CP_OEM)`. I'm not sure whether this is a valid suggestion, because the client hasn't yet installed Unicode support so it only is capable of displaying 8-bit text, and using `CP_OEM` will use the OEM code page on the server, which doesn't necessarily match the OEM code page on the client.
>
> What is the correct way of generating the error message string?

Now, mind you, the argument against using `CP_OEM` is the same argument against using `FormatMessageA`! In neither case are you sure that the code page on the server matches the code page on the client. If `CP_OEM` is wrong, then so too is `FormatMessageA` (which uses `CP_ACP`). The correct solution is to use `FormatMessageW` followed by `WideCharToMultiByte(x)`, where `x` is the OEM code page of the client. You need to get this information from the client to the server somehow so that the server knows what character set the client is going to use for displaying strings.

There's really nothing deep going on here. If you're going to display an 8-bit string, you need to use the same code page when generating the string as you will use when displaying it. Keep your eye on the code page.

Raymond Chen

Follow