How to fill in that number grouping member of NUMBERFMT

devblogs.microsoft.com/oldnewthing/20060418-11

April 18, 2006

Raymond Chen

If you look at the `NUMBERFMT` structure, the way groups are expressed by the `Grouping` member do not match the value returned by `LOCALE_SGROUPING`:

LOCALE_ SGROUPING	Grouping	Sample	Culture
3;0	3	1,234,567	United States
3;2;0	32	12,34,567	India
3	30	1234,567	(none I know of)

`LOCALE_SGROUPING` expresses grouping as a series of semicolon-separated numbers, each expressing the number of digits in each group (least-significant group first). A trailing zero indicates that the last grouping should be repeated indefinitely. For example, "3;2;0" means "Group the three least significant digits, then in twos until you run out of digits." If there is no trailing ";0", then there are no commas past that point. For example, "3" means "Group the three least significant digits, then stop." The `Grouping` member expresses the grouping rules differently. Each significant digit represents a group, with the most significant digit representing the least-significant group, with the units digit repeated indefinitely. For example, "32" means "make a group of three digits, then group by twos thereafter." To suppress the repetition, multiply by ten. In other words, the two systems are basically the same, with the `Grouping` consisting of the `LOCALE_SGROUPING` string with the semicolons removed. Except that the meaning of the trailing zero is reversed, so if `LOCALE_SGROUPING` has a trailing zero, you have to remove it to get the `Grouping`, and if it lacks a trailing zero, then you have to add one to the `Grouping`. It's kind of strange that the two systems differ, considering that they both came from the same NLS team! It's probably a case of parallel evolution, wherein the locale-string folks and the number-formatting folks came up with their respective systems independently. Writing code to implement this conversion from `LOCALE_SGROUPING` to `Grouping` shouldn't be hard once you understand the algorithm, so I'll leave that as an exercise.

Fortunately, in real life you rarely have need to perform this conversion, for you can just pass the desired locale as the first parameter to the `GetNumberFormat` (or even better, `LOCALE_USER_DEFAULT`), pass a `NULL` pointer as the `lpNumberFormat`, and let NLS do all the work.

Raymond Chen

Follow