What does the minus sign in indirect localized string resources mean?

devblogs.microsoft.com/oldnewthing/20120223-00

February 23, 2012

Raymond Chen

The syntax for indirect localized string resources is `@`*filename*`,-`*stringId*, optionally followed by a semicolon and a comment. A customer wanted to know what the minus signs stands for. The minus sign doesn't "stand for" anything. It's just part of the syntax. It's like asking what the semicolon at the end of a C statement stands for. It doesn't stand for anything; it's just part of the rules for C statements. (And if the minus sign has to stand for something, what does the comma stand for?) Okay, so maybe the question was really "Why does the syntax for indirect localized strings include a minus sign? Isn't the comma enough?" From a parsing standpoint, the comma is enough. The syntax for indirect strings was influenced by the syntax for icon locations, which also takes the form *filename*`,` *number*. We saw some time ago that the number after the comma can be positive or negative or zero. If positive or zero, it specifies the zero-based icon index. If negative, then it specifies the (negative of the) icon ID. The indirect string syntax follows the same pattern, except that they don't support string indices. (As we saw earlier when we studied the format of string resources, a null string is indistinguishable from no string at all, which makes string indices largely meaningless since you can't tell whether a null string should be counted towards the index or not.) Since the only thing supported is IDs, and IDs are expressed as negative values, the first thing after the comma is always a minus sign.

Next time, we'll take a closer look at that comment field.

Raymond Chen

Follow