Why are timer IDs and dialog control IDs 64-bit values on 64-bit Windows? Did you really expect people to create more than 4 billion timers or dialog controls?

devblogs.microsoft.com/oldnewthing/20191010-00

October 10, 2019

Raymond Chen

In 32-bit Windows, timer IDs and dialog control IDs are 32-bit values, but in 64-bit windows, timer IDs and dialog control IDs are 64-bit values. Seriously? Did you really expect people to create more than 4 billion timers or dialog controls?

No, that's not why they are 64-bit values.

Values were expanded from 32-bit values to 64-bit values whenever they could be used to store a pointer, because pointers on 64-bit Windows are 64 bits in size.

Sometimes this was obvious, like the message `WPARAM` and `LPARAM`, which are often window handles or pointers to supplementary structures. Or values used as reference data for callbacks, since a common pattern is to use a pointer to a helper structure in order to pass a large amount of information. The `GWLP_USERDATA` storage that comes with every window is another place where it is common practice to store a pointer.

We saw last time that a common convention is to use pointers to assign unique IDs, particularly for timers, and that meant that timer IDs needed to be expanded from 32-bit values to 64-bit values.

Dialog controls are stored in the `GWL_ID` window data. Why was this expanded to a 64-bit value `GWLP_ID`? You can't actually use a 64-bit dialog control ID because the `GetDlgCtrlID` function returns a 32-bit integer, and window messages that use the dialog control ID (such as `WM_COMMAND`) have only a 16-bit field for the dialog control ID. So it seems that there's no point in expanding the dialog control ID to a 64-bit value, since nobody actually reports it as a 64-bit value.

While it's true that nobody reports it as a 64-bit value, you can still retrieve the full 64-bit value by asking for the `GWLP_ID`.

Pointers are like weeds. Anywhere it's possible to fit a pointer, a pointer will try to squeeze in there.

Since it was possible in 32-bit Windows to store a 32-bit value in the place that holds the dialog control ID, people used it to store a pointer because *hey, look, free pointer-sized memory*. They never intended to use it as a dialog control ID, but if somebody's going to give you free memory, why not use it!

This creative use of the dialog control ID were discovered during the development of 64-bit Windows, and the dialog control ID storage was expanded to a 64-bit value to accommodate it.

Raymond Chen

Follow