SetWindowLongA function (winuser.h) - Win32 apps

By jwmsft

Archived: 2026-04-05 22:27:55 UTC

Changes an attribute of the specified window. The function also sets the 32-bit (long) value at the specified offset into the extra window memory.

Note This function has been superseded by the SetWindowLongPtr function. To write code that is compatible with both 32-bit and 64-bit versions of Windows, use the **SetWindowLongPtr** function.

```
 LONG SetWindowLongA(
  [in] HWND hWnd,
  [in] int  nIndex,
  [in] LONG dwNewLong
);
```

`[in] hWnd`

Type: **HWND**

A handle to the window and, indirectly, the class to which the window belongs.

`[in] nIndex`

Type: **int**

The zero-based offset to the value to be set. Valid values are in the range zero through the number of bytes of extra window memory, minus the size of an integer. To set any other value, specify one of the following values.

Value	Meaning
GWL_EXSTYLE -20	Sets a new extended window style.
GWL_HINSTANCE -6	Sets a new application instance handle.
GWL_HWNDPARENT -8	Sets a new owner for a top-level window.

GWL_ID -12	Sets a new identifier of the child window. The window cannot be a top-level window.
GWL_STYLE -16	Sets a new window style.
GWL_USERDATA -21	Sets the user data associated with the window. This data is intended for use by the application that created the window. Its value is initially zero.
GWL_WNDPROC -4	Sets a new address for the window procedure. You cannot change this attribute if the window does not belong to the same process as the calling thread.

The following values are also available when the *hWnd* parameter identifies a dialog box.

Value	Meaning
DWL_DLGPROC DWLP_MSGRESULT + sizeof(LRESULT)	Sets the new address of the dialog box procedure.
DWL_MSGRESULT 0	Sets the return value of a message processed in the dialog box procedure.
DWL_USER DWLP_DLGPROC + sizeof(DLGPROC)	Sets new extra information that is private to the application, such as handles or pointers.

`[in] dwNewLong`

Type: **LONG**

The replacement value.

Type: **LONG**

If the function succeeds, the return value is the previous value of the specified 32-bit integer.

If the function fails, the return value is zero. To get extended error information, call GetLastError.

If the previous value of the specified 32-bit integer is zero, and the function succeeds, the return value is zero, but the function does not clear the last error information. This makes it difficult to determine success or failure. To deal with this, you should clear the last error information by calling SetLastError with 0 before calling **SetWindowLong**. Then, function failure will be indicated by a return value of zero and a GetLastError result that is nonzero.

Certain window data is cached, so changes you make using **SetWindowLong** will not take effect until you call the SetWindowPos function. Specifically, if you change any of the frame styles, you must call **SetWindowPos** with the **SWP_FRAMECHANGED** flag for the cache to be updated properly.

If you use **SetWindowLong** with the **GWL_WNDPROC** index to replace the window procedure, the window procedure must conform to the guidelines specified in the description of the WindowProc callback function.

If you use **SetWindowLong** with the **DWL_MSGRESULT** index to set the return value for a message processed by a dialog procedure, you should return **TRUE** directly afterward. Otherwise, if you call any function that results in your dialog procedure receiving a window message, the nested window message could overwrite the return value you set using **DWL_MSGRESULT**.

Calling **SetWindowLong** with the **GWL_WNDPROC** index creates a subclass of the window class used to create the window. An application can subclass a system class, but should not subclass a window class created by another process. The **SetWindowLong** function creates the window subclass by changing the window procedure associated with a particular window class, causing the system to call the new window procedure instead of the previous one. An application must pass any messages not processed by the new window procedure to the previous window procedure by calling CallWindowProc. This allows the application to create a chain of window procedures.

Reserve extra window memory by specifying a nonzero value in the **cbWndExtra** member of the WNDCLASSEX structure used with the RegisterClassEx function.

Do not call **SetWindowLong** with the **GWL_HWNDPARENT** index to change the parent of a child window. Instead, use the SetParent function.

GWL_HWNDPARENT is used to change the owner of a top-level window, not the parent of a child window.

A window can have either a parent or an owner, or neither, but never both simultaneously.

If the window has a class style of **CS_CLASSDC** or **CS_OWNDC**, do not set the extended window styles **WS_EX_COMPOSITED** or **WS_EX_LAYERED**.

Calling **SetWindowLong** to set the style on a progressbar will reset its position.

For an example, see Subclassing a Window.

Note

The winuser.h header defines SetWindowLong as an alias that automatically selects the ANSI or Unicode version of this function based on the definition of the UNICODE preprocessor constant. Mixing usage of the encoding-neutral alias with code that is not encoding-neutral can lead to mismatches that result in compilation or runtime errors. For more information, see Conventions for Function Prototypes.

Requirement	Value
Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Target Platform	Windows
Header	winuser.h (include Windows.h)
Library	User32.lib
DLL	User32.dll
API set	ext-ms-win-ntuser-windowclass-l1-1-0 (introduced in Windows 8)

CallWindowProc

Conceptual

GetWindowLong

Reference

RegisterClassEx

SetParent

SetWindowLongPtr

WNDCLASSEX

Window Classes

WindowProc
