|  |  |  |
| --- | --- | --- |
| **WM\_CHAR** | 0x102 | The WM\_CHAR message is posted to the window with the keyboard focus when a WM\_KEYDOWN message is translated by the [TranslateMessage](http://pinvoke.net/default.aspx/user32/TranslateMessage.html) function. The WM\_CHAR message contains the character code of the key that was pressed. |
| **WM\_CHARTOITEM** | 0x2F | Sent by a list box with the LBS\_WANTKEYBOARDINPUT style to its owner in response to a WM\_CHAR message. |
| **WM\_CLEAR** | 0x303 | An application sends a WM\_CLEAR message to an edit control or combo box to delete (clear) the current selection, if any, from the edit control. |
| **WM\_CLOSE** | 0x10 | The WM\_CLOSE message is sent as a signal that a window or an application should terminate. |
| **WM\_COMMAND** | 0x111 | The WM\_COMMAND message is sent when the user selects a command item from a menu, when a control sends a notification message to its parent window, or when an accelerator keystroke is translated. |
| **WM\_CONTEXTMENU** | 0x7B | The WM\_CONTEXTMENU message notifies a window that the user clicked the right mouse button (right-clicked) in the window. |
| **WM\_CREATE** | 0x1 | The WM\_CREATE message is sent when an application requests that a window be created by calling the [CreateWindowEx](http://pinvoke.net/default.aspx/user32/CreateWindowEx.html) or [CreateWindow](http://pinvoke.net/default.aspx/user32/CreateWindow.html) function. (The message is sent before the function returns.) The window procedure of the new window receives this message after the window is created, but before the window becomes visible. |
| **WM\_CTLCOLORBTN** | 0x135 | The WM\_CTLCOLORBTN message is sent to the parent window of a button before drawing the button. The parent window can change the button's text and background colors. However, only owner-drawn buttons respond to the parent window processing this message. |
| **WM\_CTLCOLORDLG** | 0x136 | The WM\_CTLCOLORDLG message is sent to a dialog box before the system draws the dialog box. By responding to this message, the dialog box can set its text and background colors using the specified display device context handle. |
| **WM\_CTLCOLOREDIT** | 0x133 | An edit control that is not read-only or disabled sends the WM\_CTLCOLOREDIT message to its parent window when the control is about to be drawn. By responding to this message, the parent window can use the specified device context handle to set the text and background colors of the edit control. |
| **WM\_CTLCOLORLISTBOX** | 0x134 | Sent to the parent window of a list box before the system draws the list box. By responding to this message, the parent window can set the text and background colors of the list box by using the specified display device context handle. |
| **WM\_CTLCOLORMSGBOX** | 0x132 | The WM\_CTLCOLORMSGBOX message is sent to the owner window of a message box before Windows draws the message box. By responding to this message, the owner window can set the text and background colors of the message box by using the given display device context handle. |
| **WM\_CTLCOLORSCROLLBAR** | 0x137 | The WM\_CTLCOLORSCROLLBAR message is sent to the parent window of a scroll bar control when the control is about to be drawn. By responding to this message, the parent window can use the display context handle to set the background color of the scroll bar control. |
| **WM\_CTLCOLORSTATIC** | 0x138 | A static control, or an edit control that is read-only or disabled, sends the WM\_CTLCOLORSTATIC message to its parent window when the control is about to be drawn. By responding to this message, the parent window can use the specified device context handle to set the text and background colors of the static control. |
| **WM\_DESTROY** | 0x2 | The WM\_DESTROY message is sent when a window is being destroyed. It is sent to the window procedure of the window being destroyed after the window is removed from the screen. This message is sent first to the window being destroyed and then to the child windows (if any) as they are destroyed. During the processing of the message, it can be assumed that all child windows still exist. |
| **WM\_DISPLAYCHANGE** | 0x7E | The WM\_DISPLAYCHANGE message is sent to all windows when the display resolution has changed. |
| **WM\_DROPFILES** | 0x233 | Sent when the user drops a file on the window of an application that has registered itself as a recipient of dropped files. |
| **WM\_ENABLE** | 0xA | The WM\_ENABLE message is sent when an application changes the enabled state of a window. It is sent to the window whose enabled state is changing. This message is sent before the [EnableWindow](http://pinvoke.net/default.aspx/user32/EnableWindow.html) function returns, but after the enabled state (WS\_DISABLED style bit) of the window has changed. |
| **WM\_GETHOTKEY** | 0x33 | An application sends a WM\_GETHOTKEY message to determine the hot key associated with a window. |
| **WM\_GETICON** | 0x7F | The WM\_GETICON message is sent to a window to retrieve a handle to the large or small icon associated with a window. The system displays the large icon in the ALT+TAB dialog, and the small icon in the window caption. |
| **WM\_GETMINMAXINFO** | 0x24 | The WM\_GETMINMAXINFO message is sent to a window when the size or position of the window is about to change. An application can use this message to override the window's default maximized size and position, or its default minimum or maximum tracking size. |
| **WM\_HELP** | 0x53 | Indicates that the user pressed the F1 key. If a menu is active when F1 is pressed, WM\_HELP is sent to the window associated with the menu; otherwise, WM\_HELP is sent to the window that has the keyboard focus. If no window has the keyboard focus, WM\_HELP is sent to the currently active window. |
| **WM\_HOTKEY** | 0x312 | The WM\_HOTKEY message is posted when the user presses a hot key registered by the [RegisterHotKey](http://pinvoke.net/default.aspx/user32/RegisterHotKey.html) function. The message is placed at the top of the message queue associated with the thread that registered the hot key. |
| **WM\_HSCROLL** | 0x114 | This message is sent to a window when a scroll event occurs in the window's standard horizontal scroll bar. This message is also sent to the owner of a horizontal scroll bar control when a scroll event occurs in the control. |
| **WM\_INITMENU** | 0x116 | The WM\_INITMENU message is sent when a menu is about to become active. It occurs when the user clicks an item on the menu bar or presses a menu key. This allows the application to modify the menu before it is displayed. |
| **WM\_INITMENUPOPUP** | 0x117 | The WM\_INITMENUPOPUP message is sent when a drop-down menu or submenu is about to become active. This allows an application to modify the menu before it is displayed, without changing the entire menu. |
| **WM\_KEYDOWN** | 0x100 | The WM\_KEYDOWN message is posted to the window with the keyboard focus when a nonsystem key is pressed. A nonsystem key is a key that is pressed when the ALT key is not pressed. |
| **WM\_KEYUP** | 0x101 | The WM\_KEYUP message is posted to the window with the keyboard focus when a nonsystem key is released. A nonsystem key is a key that is pressed when the ALT key is not pressed, or a keyboard key that is pressed when a window has the keyboard focus. |
| **WM\_LBUTTONDBLCLK** | 0x203 | The WM\_LBUTTONDBLCLK message is posted when the user double-clicks the left mouse button while the cursor is in the client area of a window. If the mouse is not captured, the message is posted to the window beneath the cursor. Otherwise, the message is posted to the window that has captured the mouse. |
| **WM\_LBUTTONDOWN** | 0x201 | The WM\_LBUTTONDOWN message is posted when the user presses the left mouse button while the cursor is in the client area of a window. If the mouse is not captured, the message is posted to the window beneath the cursor. Otherwise, the message is posted to the window that has captured the mouse. |
| **WM\_LBUTTONUP** | 0x202 | The WM\_LBUTTONUP message is posted when the user releases the left mouse button while the cursor is in the client area of a window. If the mouse is not captured, the message is posted to the window beneath the cursor. Otherwise, the message is posted to the window that has captured the mouse. |
| **WM\_MBUTTONDBLCLK** | 0x209 | The WM\_MBUTTONDBLCLK message is posted when the user double-clicks the middle mouse button while the cursor is in the client area of a window. If the mouse is not captured, the message is posted to the window beneath the cursor. Otherwise, the message is posted to the window that has captured the mouse. |
| **WM\_MBUTTONDOWN** | 0x207 | The WM\_MBUTTONDOWN message is posted when the user presses the middle mouse button while the cursor is in the client area of a window. If the mouse is not captured, the message is posted to the window beneath the cursor. Otherwise, the message is posted to the window that has captured the mouse. |
| **WM\_MBUTTONUP** | 0x208 | The WM\_MBUTTONUP message is posted when the user releases the middle mouse button while the cursor is in the client area of a window. If the mouse is not captured, the message is posted to the window beneath the cursor. Otherwise, the message is posted to the window that has captured the mouse. |
| **WM\_MOUSEACTIVATE** | 0x21 | The WM\_MOUSEACTIVATE message is sent when the cursor is in an inactive window and the user presses a mouse button. The parent window receives this message only if the child window passes it to the [DefWindowProc](http://pinvoke.net/default.aspx/user32/DefWindowProc.html) function. |
| **WM\_MOUSEHOVER** | 0x2A1 | The WM\_MOUSEHOVER message is posted to a window when the cursor hovers over the client area of the window for the period of time specified in a prior call toTrackMouseEvent. |
| **WM\_MOUSELEAVE** | 0x2A3 | The WM\_MOUSELEAVE message is posted to a window when the cursor leaves the client area of the window specified in a prior call to TrackMouseEvent. |
| **WM\_MOUSEMOVE** | 0x200 | The WM\_MOUSEMOVE message is posted to a window when the cursor moves. If the mouse is not captured, the message is posted to the window that contains the cursor. Otherwise, the message is posted to the window that has captured the mouse. |
| **WM\_MOVE** | 0x3 | The WM\_MOVE message is sent after a window has been moved. |
| **WM\_MOVING** | 0x216 | The WM\_MOVING message is sent to a window that the user is moving. By processing this message, an application can monitor the position of the drag rectangle and, if needed, change its position. |
| **WM\_NOTIFY** | 0x4E | Sent by a common control to its parent window when an event has occurred or the control requires some information. |
| **WM\_NOTIFYFORMAT** | 0x55 | Determines if a window accepts ANSI or Unicode structures in the WM\_NOTIFY notification message. WM\_NOTIFYFORMAT messages are sent from a common control to its parent window and from the parent window to the common control. |
| **WM\_NULL** | 0x0 | The WM\_NULL message performs no operation. An application sends the WM\_NULL message if it wants to post a message that the recipient window will ignore. |
| **WM\_PAINT** | 0xF | Occurs when the control needs repainting |
| **WM\_PAINTICON** | 0x26 | Windows NT 3.51 and earlier: The WM\_PAINTICON message is sent to a minimized window when the icon is to be painted. This message is not sent by newer versions of Microsoft Windows, except in unusual circumstances explained in the Remarks. |
| **WM\_POWERBROADCAST** | 0x218 | Notifies applications that a power-management event has occurred. |
| **WM\_PRINT** | 0x317 | The WM\_PRINT message is sent to a window to request that it draw itself in the specified device context, most commonly in a printer device context. |
| **WM\_PRINTCLIENT** | 0x318 | The WM\_PRINTCLIENT message is sent to a window to request that it draw its client area in the specified device context, most commonly in a printer device context. |
| **WM\_QUIT** | 0x12 | Once received, it ends the application's Message Loop, signaling the application to end. It can be sent by pressing Alt+F4, Clicking the X in the upper right-hand of the program, or going to File->Exit. |
| **WM\_RBUTTONDBLCLK** | 0x206 | he WM\_RBUTTONDBLCLK message is posted when the user double-clicks the right mouse button while the cursor is in the client area of a window. If the mouse is not captured, the message is posted to the window beneath the cursor. Otherwise, the message is posted to the window that has captured the mouse. |
| **WM\_RBUTTONDOWN** | 0x204 | The WM\_RBUTTONDOWN message is posted when the user presses the right mouse button while the cursor is in the client area of a window. If the mouse is not captured, the message is posted to the window beneath the cursor. Otherwise, the message is posted to the window that has captured the mouse. |
| **WM\_RBUTTONUP** | 0x205 | The WM\_RBUTTONUP message is posted when the user releases the right mouse button while the cursor is in the client area of a window. If the mouse is not captured, the message is posted to the window beneath the cursor. Otherwise, the message is posted to the window that has captured the mouse. |
| **WM\_SETCURSOR** | 0x20 | The WM\_SETCURSOR message is sent to a window if the mouse causes the cursor to move within a window and mouse input is not captured. |
| **WM\_SETFOCUS** | 0x7 | When the controll got the focus |
| **WM\_SETHOTKEY** | 0x32 | An application sends a WM\_SETHOTKEY message to a window to associate a hot key with the window. When the user presses the hot key, the system activates the window. |
| **WM\_SHOWWINDOW** | 0x18 | The WM\_SHOWWINDOW message is sent to a window when the window is about to be hidden or shown |
| **WM\_SIZE** | 0x5 | The WM\_SIZE message is sent to a window after its size has changed. |
| **WM\_SIZING** | 0x214 | The WM\_SIZING message is sent to a window that the user is resizing. By processing this message, an application can monitor the size and position of the drag rectangle and, if needed, change its size or position. |
| **WM\_SYSCHAR** | 0x106 | The WM\_SYSCHAR message is posted to the window with the keyboard focus when a WM\_SYSKEYDOWN message is translated by the [TranslateMessage](http://pinvoke.net/default.aspx/user32/TranslateMessage.html) function. It specifies the character code of a system character key — that is, a character key that is pressed while the ALT key is down. |
| **WM\_SYSCOMMAND** | 0x112 | A window receives this message when the user chooses a command from the Window menu (formerly known as the system or control menu) or when the user chooses the maximize button, minimize button, restore button, or close button. |
| **WM\_SYSDEADCHAR** | 0x107 | The WM\_SYSDEADCHAR message is sent to the window with the keyboard focus when a WM\_SYSKEYDOWN message is translated by the [TranslateMessage](http://pinvoke.net/default.aspx/user32/TranslateMessage.html)function. WM\_SYSDEADCHAR specifies the character code of a system dead key — that is, a dead key that is pressed while holding down the ALT key. |
| **WM\_SYSKEYDOWN** | 0x104 | The WM\_SYSKEYDOWN message is posted to the window with the keyboard focus when the user presses the F10 key (which activates the menu bar) or holds down the ALT key and then presses another key. It also occurs when no window currently has the keyboard focus; in this case, the WM\_SYSKEYDOWN message is sent to the active window. The window that receives the message can distinguish between these two contexts by checking the context code in the lParam parameter. |
| **WM\_SYSKEYUP** | 0x105 | The WM\_SYSKEYUP message is posted to the window with the keyboard focus when the user releases a key that was pressed while the ALT key was held down. It also occurs when no window currently has the keyboard focus; in this case, the WM\_SYSKEYUP message is sent to the active window. The window that receives the message can distinguish between these two contexts by checking the context code in the lParam parameter. |
| **WM\_TIMER** | 0x113 | The WM\_TIMER message is posted to the installing thread's message queue when a timer expires. The message is posted by the [GetMessage](http://pinvoke.net/default.aspx/user32/GetMessage.html) or [PeekMessage](http://pinvoke.net/default.aspx/user32/PeekMessage.html)function. |
| **WM\_VSCROLL** | 0x115 | The WM\_VSCROLL message is sent to a window when a scroll event occurs in the window's standard vertical scroll bar. This message is also sent to the owner of a vertical scroll bar control when a scroll event occurs in the control. |
| **WM\_WINDOWPOSCHANGED** | 0x47 | The WM\_WINDOWPOSCHANGED message is sent to a window whose size, position, or place in the Z order has changed as a result of a call to the SetWindowPosfunction or another window-management function. |
| **WM\_WINDOWPOSCHANGING** | 0x46 | The WM\_WINDOWPOSCHANGING message is sent to a window whose size, position, or place in the Z order is about to change as a result of a call to theSetWindowPos function or another window-management function. |
| **WM\_XBUTTONDBLCLK** | 0x20D | The **WM\_XBUTTONDBLCLK** message is posted when the user double-clicks the first or second X button while the cursor is in the client area of a window. If the mouse is not captured, the message is posted to the window beneath the cursor. Otherwise, the message is posted to the window that has captured the mouse. |
| **WM\_XBUTTONDOWN** | 0x20B | The **WM\_XBUTTONDOWN** message is posted when the user presses the first or second X button while the cursor is in the client area of a window. If the mouse is not captured, the message is posted to the window beneath the cursor. Otherwise, the message is posted to the window that has captured the mouse. |
| **WM\_XBUTTONUP** | 0x20C | The **WM\_XBUTTONUP** message is posted when the user releases the first or second X button while the cursor is in the client area of a window. If the mouse is not captured, the message is posted to the window beneath the cursor. Otherwise, the message is posted to the window that has captured the mouse. |