This technical note provides background information on the new **WM\_NOTIFY** message and describes the recommended (and most common) way of handling **WM\_NOTIFY** messages in your MFC application.

**Notification Messages in Windows 3.x**

In Windows 3.x, controls notify their parents of events such as mouse clicks, changes in content and selection, and control background painting by sending a message to the parent. Simple notifications are sent as special **WM\_COMMAND** messages, with the notification code (such as **BN\_CLICKED**) and control ID packed into wParam and the control's handle in lParam. Note that since wParam and lParam are full, there is no way to pass any additional data — these messages can be only simple notification. For instance, in the **BN\_CLICKED** notification, there's no way to send information about the location of the mouse cursor when the button was clicked.

When controls in Windows 3.x need to send a notification message that includes additional data, they use a variety of special-purpose messages, including **WM\_CTLCOLOR**, **WM\_VSCROLL**, **WM\_HSCROLL**, **WM\_DRAWITEM**, **WM\_MEASUREITEM**, **WM\_COMPAREITEM**, **WM\_DELETEITEM**, **WM\_CHARTOITEM**, **WM\_VKEYTOITEM**, and so on. These messages can be reflected back to the control that sent them. For more information, see [TN062: Message Reflection for Windows Controls](http://msdn.microsoft.com/en-us/library/eeah46xd.aspx).

**Notification Messages in Win32**

For controls that existed in Windows 3.1, the Win32 API uses most of the notification messages that were used in Windows 3.x. However, Win32 also adds a number of sophisticated, complex controls to those supported in Windows 3.x. Frequently, these controls need to send additional data with their notification messages. Rather than adding a new **WM\_\*** message for each new notification that needs additional data, the designers of the Win32 API chose to add just one message, **WM\_NOTIFY**, which can pass any amount of additional data in a standardized fashion.

**WM\_NOTIFY** messages contain the ID of the control sending the message in wParam and a pointer to a structure in lParam. This structure is either an **NMHDR** structure or some larger structure that has an **NMHDR** structure as its first member. Note that since the **NMHDR** member is first, a pointer to this structure can be used as either a pointer to an **NMHDR** or as a pointer to the larger structure depending on how you cast it.

In most cases, the pointer will point to a larger structure and you'll need to cast it when you use it. In only a few notifications, such as the common notifications (whose names start with **NM\_**) and the tool tip control's **TTN\_SHOW** and **TTN\_POP** notifications, is an **NMHDR** structure actually used.

The **NMHDR(notify message header)** structure or initial member contains the handle and ID of the control sending the message and the notification code (such as **TTN\_SHOW**). The format of the **NMHDR** structure is shown below:

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_f313b26f-4c32-4f69-ad7f-1c148acc2c08');" \o "Copy to clipboard.)

typedef struct tagNMHDR {

HWND hwndFrom;

UINT idFrom;

UINT code;

} NMHDR;

For a **TTN\_SHOW** message, the **code** member would be set to **TTN\_SHOW**.

Most notifications pass a pointer to a larger structure that contains an **NMHDR** structure as its first member. For instance, consider the structure used by the list view control's **LVN\_KEYDOWN** notification message, which is sent when a key is pressed in a list view control. The pointer points to an **LV\_KEYDOWN** structure, which is defined as shown below:

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_faedb6f3-932d-4fd1-8bde-5720be8f0af6');)

typedef struct tagLV\_KEYDOWN {

NMHDR hdr;

WORD wVKey;

UINT flags;

} LV\_KEYDOWN;

Note that since the **NMHDR** member is first in this structure, the pointer you're passed in the notification message can be cast to either a pointer to an **NMHDR** or a pointer to an **LV\_KEYDOWN**.

**Notifications Common to All New Windows Controls**

Some notifications are common to all of the new Windows controls. These notifications pass a pointer to an **NMHDR** structure.

|  |  |
| --- | --- |
| **Notification code** | **Sent because** |
| **NM\_CLICK** | User clicked left mouse button in the control |
| **NM\_DBLCLK** | User double-clicked left mouse button in the control |
| **NM\_RCLICK** | User clicked right mouse button in the control |
| **NM\_RDBLCLK** | User double-clicked right mouse button in the control |
| **NM\_RETURN** | User pressed the ENTER key while control has input focus |
| **NM\_SETFOCUS** | Control has been given input focus |
| **NM\_KILLFOCUS** | Control has lost input focus |
| **NM\_OUTOFMEMORY** | Control could not complete an operation because there was not enough memory available |

[**ON\_NOTIFY: Handling WM\_NOTIFY Messages in MFC Applications**](javascript:void(0))

The function **CWnd::OnNotify** handles notification messages. Its default implementation checks the message map for notification handlers to call. In general, you do not override **OnNotify**. Instead, you provide a handler function and add a message-map entry for that handler to the message map of your owner window's class.

ClassWizard, via the ClassWizard property sheet, can create the **ON\_NOTIFY** message-map entry and provide you with a skeleton handler function. For more information on using ClassWizard to make this easier, see [Mapping Messages to Functions](http://msdn.microsoft.com/en-us/library/dey7ke4c.aspx).

The **ON\_NOTIFY** message-map macro has the following syntax:

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_28b686fb-2447-4926-beea-163144aa5edc');)

ON\_NOTIFY( wNotifyCode, id, memberFxn )

where the italicized parameters are replaced with:

wNotifyCode

The code for the notification message to be handled, such as **LVN\_KEYDOWN**.

id

The child identifier of the control for which the notification is sent.

memberFxn

The member function to be called when this notification is sent.

Your member function must be declared with the following prototype:

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_25713dc3-2451-46d7-8d84-a10fb6ad6be3');)

afx\_msg void memberFxn( NMHDR \* pNotifyStruct, LRESULT \* result );

[**Remarks**](javascript:void(0))

where the italicized parameters are:

pNotifyStruct

A pointer to the notification structure, as described in the section above.

*result*

A pointer to the result code you'll set before you return.

[**Example**](javascript:void(0))

To specify that you want the member function OnKeydownList1 to handle **LVN\_KEYDOWN** messages from the **CListCtrl** whose ID is IDC\_LIST1, you would use ClassWizard to add the following to your message map:

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_55b91792-eb64-4df4-aebf-11b9e533b7df');)

ON\_NOTIFY( LVN\_KEYDOWN, IDC\_LIST1, OnKeydownList1 )

In the example above, the function provided by ClassWizard is:

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_0fe65e7a-8bd4-4661-a00a-66a6720eae8f');)

void CMessageReflectionDlg::OnKeydownList1(NMHDR\* pNMHDR, LRESULT\* pResult)

{

LV\_KEYDOWN\* pLVKeyDow = (LV\_KEYDOWN\*)pNMHDR;

// TODO: Add your control notification handler

// code here

\*pResult = 0;

}

Note that ClassWizard provides a pointer of the proper type automatically. You can access the notification structure through either pNMHDR or pLVKeyDow.

[**ON\_NOTIFY\_RANGE**](javascript:void(0))

If you need to process the same **WM\_NOTIFY** message for a set of controls, you can use **ON\_NOTIFY\_RANGE** rather than **ON\_NOTIFY**. For instance, you may have a set of buttons for which you want to perform the same action for a certain notification message.

When you use **ON\_NOTIFY\_RANGE**, you specify a contiguous range of child identifiers for which to handle the notification message by specifying the beginning and ending child identifiers of the range.

ClassWizard does not handle **ON\_NOTIFY\_RANGE**; to use it, you need to edit your message map yourself.

The message-map entry and function prototype for **ON\_NOTIFY\_RANGE** are as follows:

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_85f0050e-03c8-436c-9368-3aa78c83d4b0');)

ON\_NOTIFY\_RANGE( wNotifyCode, id, idLast, memberFxn )

where the italicized parameters are replaced with:

wNotifyCode

The code for the notification message to be handled, such as **LVN\_KEYDOWN**.

id

The first identifier in the contiguous range of identifiers.

idLast

The last identifier in the contiguous range of identifiers.

memberFxn

The member function to be called when this notification is sent.

Your member function must be declared with the following prototype:

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_1a8d12f8-3a14-4596-9049-30a4b4b724ea');)

afx\_msg void memberFxn( UINT id, NMHDR \* pNotifyStruct, LRESULT \* result );

[**Remarks**](javascript:void(0))

where the italicized parameters are:

id

The child identifier of the control that sent the notification.

pNotifyStruct

A pointer to the notification structure, as described above.

*result*

A pointer to the result code you'll set before you return.

[**ON\_NOTIFY\_EX, ON\_NOTIFY\_EX\_RANGE**](javascript:void(0))

If you want more than one object in the notification routing to handle a message, you can use **ON\_NOTIFY\_EX** (or **ON\_NOTIFY\_EX\_RANGE**) rather than **ON\_NOTIFY** (or **ON\_NOTIFY\_RANGE**). The only difference between the **EX** version and the regular version is that the member function called for the **EX** version returns a **BOOL** that indicates whether or not message processing should continue. Returning **FALSE** from this function allows you to process the same message in more than one object.

ClassWizard does not handle **ON\_NOTIFY\_EX** or **ON\_NOTIFY\_EX\_RANGE**; if you want to use either of them, you need to edit your message map yourself.

The message-map entry and function prototype for **ON\_NOTIFY\_EX** and **ON\_NOTIFY\_EX\_RANGE** are as follows. The meanings of the parameters are the same as for the non-**EX** versions.

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_1f684e8c-5d0e-47e1-85c3-a6506b34ac4f');)

ON\_NOTIFY\_EX( nCode, id, memberFxn )

ON\_NOTIFY\_EX\_RANGE( wNotifyCode, id, idLast, memberFxn )

The prototype for both of the above is the same:

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_e45563b2-f2d3-44c2-a0e1-ec9c538e4dfb');)

afx\_msg BOOL memberFxn( UINT id, NMHDR \* pNotifyStruct, LRESULT \* result );