VCL Component Messages

(Zusammengestellt von Mike Lischke [http://www.lischke-online.de],
bearbeitet von Simon Reinhardt [http://www.pics-software.de])

{Compiled by Mike Lischke [http://www.lischke-online.de],
edited by Simon Reinhardt [http://www.pics-software.de])

Component Messages (CM_) werden ausschließlich von der VCL generiert und sind nicht reflektierte Windows Messages (WM_), wie man annehmen könnte. Component Notifications (CN_) sind dagegen reflektierte Windows Messages.

Der Sinn ist, dass Windows oft Nachrichten an das Elternfenster eines Controls anstelle des Controls selbst versendet. Die VCL konvertiert diese einfach zu Component Notifications und sendet sie dann wieder an das Control für das die Message eigentlich bestimmt war.

<TRANSLATION>

Component Messages (CM_) are generated only by the VCL and are not reflected Windows Messages (WM_), as one may assume. In spite of that Component Notifications (CN_) are reflected Windows Messages.

The idea behind it is, that Windows often sends messages to a parent window of a control instead of the control itself. The VCL simply converts (reflects) these messages to Component Notifications and then sends it to the control, for which the message originally was meant.

</TRANSLATION>

VCL Control Messages (values):	**parameters:**	**comments:**
CM_BASE ($B000)		
CM_ACTIVATE (CM_BASE + 0)	no params	used when app is actived or a custom form is modally shown
CM_DEACTIVATE (CM_BASE + 1)	no params	counter part to CM_ACTIVATE
CM_GOTFOCUS (CM_BASE + 2)	no params	not used
CM_LOSTFOCUS (CM_BASE + 3)	no params	not used
CM_CANCELMODE (CM_BASE + 4)	TCMCancelMode	used to stop special behaviour of controls like TDBLookup
CM_DIALOGKEY (CM_BASE + 5)	TCMDialogKey	used in the KeyPreview chain of a form to determin whether a child control processes special keys like

		TAB, arrow keys etc.
CM_DIALOGCHAR (CM_BASE + 6)	TCMDialogChar	same as CM_DIALOGKEY but for characters
CM_FOCUSCHANGED (CM_BASE + 7)	TCMFocusChanged	used in forms when the active control changes
CM_PARENTFONTCHANGED (CM_BASE + 8)	if wParam = 1 then lParam contains a TFont else just use the font of the parent used in all controls	
CM_PARENTCOLORCHANGED (CM_BASE + 9)	if wParam = 1 then lParam contains a color else just use the color of the parent used in all controls	
CM_HITTEST (CM_BASE + 10)	TCMHitTest	used only at design time (and only in ControlAtPos) to determine the control at the current mouse Message.Result = 0 if control considers itself as not hit else 1
CM_VISIBLECHANGED (CM_BASE + 11)	wParam = 1 if visible, lParam contains	used when a control's visibility is changed by its Visible property

CM_ENABLEDCHANGED or (CM_BASE + 12)	no params	used when a control is enabled disabled by its Enabled property
CM_COLORCHANGED (CM_BASE + 13)	no params	used when a control's color is changed by its Color property
CM_FONTCHANGED (CM_BASE + 14)	no params	used in the TFont.OnChange event of a control's font
CM_CURSORCHANGED (CM_BASE + 15)	no params	used when a control's cursor is changed by its Cursor property
CM_CTL3DCHANGED (CM_BASE + 16)	no params	used when control's Ctrl3D property is changed
CM_PARENTCTL3DCHANGED (CM_BASE + 17) message	no params	used with ActiveX control (OnAmbientPropertyChange), in response to a CM_CTRL3DCHANGED and used when reading a TWinControl from a stream
CM_TEXTCHANGED (CM_BASE + 18)	no params	used when a control's text is changed
CM_MOUSEENTER (CM_BASE + 19)	wParam = 0, lParam = sometimes a reference of the control having the mouse pointer over it sent by TApplication and used internally by TSpeedButton	
CM_MOUSELEAVE (CM_BASE + 20)	as in CM_MOUSEENTER	counterpart to CM_MOUSEENTER
CM_MENUCHANGED (CM_BASE + 21)	no params	internal message for TMenu (when merging or an item has changed)
CM_APPKEYDOWN (CM_BASE + 22)	wParam = KeyCode, lParam = KeyData (like in WM_KEYDOWN)	sent only when determining whether a key is a menu key
CM_APPSYSCOMMAND (CM_BASE + 23) record	wParam = 0, lParam = @Message	sent in response to a WM_SYSCOMMAND message (received by a TWinControl) passing the entire Message in lParam (used in hint windows and by the application to focus itself)
CM_BUTTONPRESSED	wParam = group index,	used internally by

TSpeedButton (CM_BASE + 24)	lParam = button reference	to implement button groups
CM_SHOWINGCHANGED various (CM_BASE + 25) loading TCoolBand TMemoStrings	no params	sent by TWinControl on events (window creation, from stream, new parent assignment), used in on destroy and by on UpdateState change

CM_ENTER (CM_BASE + 26)	TCMEnter	sent by TForm when the focus changes to another child control and by TApplication when it gets the focus
CM_EXIT (CM_BASE + 27)	TCMExit	sent by TForm when the focus changes to another child
CM_DESIGNHITTEST from (CM_BASE + 28) whether	TCMDesignHittest	special message never sent within the VCL (but from a designer DLL), used only at design time to determine a control wants mouse and key input. Message.Result = 1 if the control behaves like at runtime else 0
CM_ICONCHANGED (CM_BASE + 29)	no params	sent by TApplication from its FIcon.OnChange event
CM_WANTSPECIALKEY (CM_BASE + 30) CN_KEYUP	TCMWantSpecialKey	sent by TControl in response to CN_KEYDOWN and
CM_INVOKEHELP (CM_BASE + 31)	wParam = command, lParam = Data	sent by TApplication to the main form on help invoking (nowhere used)
CM_WINDOWHOOK form (CM_BASE + 32)	wParam = 0 if hooking = 1 if unhooking lParam = @@Hook	sent by TApplication when a hooks or unhooks the main window proc
CM_RELEASE itself (CM_BASE + 33) notify	no params	sent by TForm on release of and by TFindDialog when to its redirector
CM_SHOWHINTCHANGED (CM_BASE + 34)	no params	used when a control's ShowHint property is changed
CM_PARENTSHOWHINTCHANGED (CM_BASE + 35) controls	no params	used at various places (in response to a CM_SHOWHINTCHANGED message, when reading a data from a stream etc.)
CM_SYSCOLORCHANGE (CM_BASE + 36)	no params	sent in response to a WM_SYSCOLORCHANGE message (by TWinControl)
CM_WININICHANGE	TWMWinIniChange	sent by TWinControl in

response (CM_BASE + 37)		to WM_WININICHANGE
CM_FONTCHANGE (CM_BASE + 38)	no params	sent by TWinControl in response to WM_FONTCHANGE
CM_TIMECHANGE (CM_BASE + 39)	no params	sent by TWinControl in response to WM_TIMECHANGE
CM_TABSTOPCHANGED (CM_BASE + 40)	no params	used when a control's TabStop property is changed

CM_UIACTIVATE (CM_BASE + 41)	no params	sent by TCustomForm when the active control changes and is used primarily for OLE controls
CM_UIDEACTIVATE (CM_BASE + 42)	no params	counterpart to CM_UIACTIVATE
CM_DOCWINDOWACTIVATE (CM_BASE + 43)	wParam = active, lParam = 0	sent by TCustomForm on activation/deactivation to the current OLE control if there's one
CM_CONTROLLISTCHANGE (CM_BASE + 44)	wParam = Control, lParam = 1 if inserting else 0	sent by TWinControl on inserting/removing a child control
CM_GETDATALINK (CM_BASE + 45)	no params	sent by TDBCtrlGrid to retrieve an eventual data link of a control (Message.Result returns the link if applicable)
CM_CHILDKEY (CM_BASE + 46)	TCMChildKey	sent by TWinControl in response to CN_KEYDOWN and CN_SYSKEYDOWN and by TActiveXControl when translating accelerator keys
CM_DRAG (CM_BASE + 47)	TCMDrag	sent by internal drag routines (Controls.pas) and used by TWinControl to trigger the various drag and dock events
CM_HINTSHOW (CM_BASE + 48)	wParam = 0, lParam = @THintInfo	sent by TApplication on hint activation and on mouse messages
CM_DIALOGHANDLE (CM_BASE + 49)	wParam = 0, lParam = Handle when setting wParam = 1, lParam = 0 when reading	sent by TApplication when DialogHandle is read or written to and the applications handle has not yet been created, when reading the dialog handle then Message.Result contains the new handle
CM_ISTOOLCONTROL (CM_BASE + 50)	no params	sent by TOLEForm to determine if its child controls are tool controls,

		used only by TCustomPanel
CM_RECREATEWND (CM_BASE + 51)	no params	sent by TWinControl.RecreateWnd and TFieldDataLink.UpdateRightToLeft
CM_INVALIDATE (CM_BASE + 52)	wParam = 1 if repaint is required else 0, lParam = 0	sent by TWinControl.Invalidate to notify itself and its parent of the invalidation request
CM_SYSFONTCHANGED is (CM_BASE + 53) or	no params	sent by TScreen if its IconFont changed and by TControl when reading properties from stream changing DesktopFont

CM_CONTROLCHANGE (CM_BASE + 54)	TCMControlChange	sent by TWinControl when inserting or removing a child control
CM_CHANGED (CM_BASE + 55)	wParam = 0, lParam = Self	sent by TControl.Changed
CM_DOCKCLIENT to (CM_BASE + 56)	TCMDockClient	sent by TWinControl.DockDrop determine whether docking is allowed or not
CM_UNDOCKCLIENT (CM_BASE + 57) removing a	TCMUndockClient	sent by TControl on Destroy, by TWinControl.DoUnDock and TCustomDockForm on child control
CM_FLOAT routines (CM_BASE + 58)	TCMFloat	sent by internal drag/dock to make a control floating
CM_BORDERCHANGED (CM_BASE + 59) styles	no params	sent by TWinControl when BorderWidth or BevelWidth are changed
CM_BIDIMODECHANGED (CM_BASE + 60)	no params	sent by TControl.SetBiDiMode
CM_PARENTBIDIMODECHANGED (CM_BASE + 61) when from	no params	sent at various places (in response to CM_BIDIMODECHANGED, reading a control from a stream, when reading a custom from a stream, when BiDiMode of TApplication changes and when TCustomRichEdit is created)
CM_ALLCHILDRENFLIPPED (CM_BASE + 62)	no params	sent by TWinControl.DoFlipChildren
CM_ACTIONUPDATE TContainedAction.Update (CM_BASE + 63)	wParam = 0, lParam = Action	sent by
CM_ACTIONEXECUTE TContainedAction.Execute (CM_BASE + 64)	wParam = 0, lParam = Action	sent by
CM_HINTSHOWPAUSE (CM_BASE + 65) has	wParam = 1 if hint was active lParam = time interval	sent by TApplication on mouse messages,when there's a new control under the mouse and

	when to show new hint	ShowHint set to True
CM_DOCKNOTIFICATION (CM_BASE + 66) to	TCMDocNotification	sent by TControl.SendDockNotification which is executed in response to CM_VISIBLECHANGED and WM_SETTEXT
CM_MOUSEWHEEL (CM_BASE + 67) registered TWInControl	TCMMouseWheel	general mousewheel message generated either by the mouse wheel message or WM_MOUSEWHEEL (not Win95) and sent by

VCL Control Notifications

VCL Control Notifications are just reflections of the corresponding WM_xxx messages. They are sent by a window to their parent window (not VCL but Windows). This makes in Delphi no sense as the parent knows basically nothing about its child windows and can therefore not handle them. Each of these messages contains in its lParam the handle of the child window which has sent the message. The VCL does nothing else then to add CN_BASE to the message value and sends the message then to the window which created it originally. This way windows can handle their special messages themselves.

Control Notification	Value
CN_BASE	$BC00
CN_CHAR	CN_BASE + WM_CHAR
CN_CHARTOITEM	CN_BASE + WM_CHARTOITEM
CN_COMMAND	CN_BASE + WM_COMMAND
CN_COMPAREITEM	CN_BASE + WM_COMPAREITEM
CN_CTLCOLORBTN	CN_BASE + WM_CTLCOLORBTN
CN_CTLCOLORDLG	CN_BASE + WM_CTLCOLORDLG
CN_CTLCOLOREDIT	CN_BASE + WM_CTLCOLOREDIT
CN_CTLCOLORLISTBOX	CN_BASE + WM_CTLCOLORLISTBOX
CN_CTLCOLORMSGBOX	CN_BASE + WM_CTLCOLORMSGBOX
CN_CTLCOLORSCROLLBAR	CN_BASE + WM_CTLCOLORSCROLLBAR
CN_CTLCOLORSTATIC	CN_BASE + WM_CTLCOLORSTATIC
CN_DELETEITEM	CN_BASE + WM_DELETEITEM
CN_DRAWITEM	CN_BASE + WM_DRAWITEM
CN_KEYDOWN	CN_BASE + WM_KEYDOWN
CN_KEYUP	CN_BASE + WM_KEYUP
CN_HSCROLL	CN_BASE + WM_HSCROLL
CN_MEASUREITEM	CN_BASE + WM_MEASUREITEM
CN_NOTIFY	CN_BASE + WM_NOTIFY
CN_PARENTNOTIFY	CN_BASE + WM_PARENTNOTIFY
CN_SYSKEYDOWN	CN_BASE + WM_SYSKEYDOWN
CN_SYSCHAR	CN_BASE + WM_SYSCHAR
CN_VKEYTOITEM	CN_BASE + WM_VKEYTOITEM
CN_VSCROLL	CN_BASE + WM_VSCROLL